DISCRETE OPTIMIZATION PROBLEMS UNDER RANKING-BASED CHOICE MODELS

A Dissertation

Presented to the Faculty of the Graduate School

of Cornell University

in Partial Fulfillment of the Requirements for the Degree of

Doctor of Philosophy

by

Alice Joanna Paul

August 2017

DISCRETE OPTIMIZATION PROBLEMS UNDER RANKING-BASED CHOICE MODELS

Alice Joanna Paul, Ph.D.

Cornell University 2017

We consider revenue management problems when customers purchase products according to a nonparametric choice model. In the nonparametric choice model, each customer arrives with a preference list and will purchase the highest-ranking offered product in her preference list. This choice model has attracted a lot of attention recently as a broad alternative to parametric models. However, the corresponding revenue management problems under this general model are intractable. This thesis introduces three models that are restrictions of the full nonparametric choice model and correspond to practical settings in which a retailer may want to use the nonparametric choice model. The models we introduce are simple to describe, easy to estimate, and admit tractable and profitable solutions.

First, we introduce the nonparametric tree choice model. In this model, we assume the set of customer classes is derived from paths in a tree, in which the order of nodes visited along each path gives the corresponding preference list. To start with, we study assortment problems, in which the goal is to find which products to offer to maximize expected revenue. We give a dynamic programming solution which can be extended to versions of the assortment problem when there are fixed costs for offering a product, shelf constraints, or substitution costs. We then study the joint assortment and pricing problem, in which the goal is to simultaneously select the set of offered products as well as their prices. We solve the pricing problem optimally when the products are vertically differentiated, and hence the

tree takes the form of a single path. We also solve the problem optimally on the general tree when the prices are restricted to be quality consistent; higher quality products must be priced above lower quality products. Last, we present computational experiments on both synthetic data and real hotel purchase data. Our estimation procedure shows both how to build the tree of products and how to estimate the underlying arrival probabilities of each customer type from historical sales data. These experiments show that the nonparametric tree choice model captures customer purchasing behavior more accurately than the multinomial logit choice model in the majority of test cases.

Second, we study a customer choice model that captures purchasing behavior when customers substitute a limited number of times. Again, under this model, we assume each customer is characterized by a ranked preference list of products and will purchase the highest ranking product in her list that is offered. Since we restrict ourselves to settings with limited substitution, we assume that these preference lists contain at most k products. We call this model the k-product nonparametric choice model. We focus on the assortment optimization problem under this choice model. First, we show that this problem is NP-hard even for $k = 2$. Motivated by this result, we show that the assortment problem under the 2-product nonparametric choice model can be reduced to a well-known graph optimization problem. Further, we develop a novel linear programming based rounding algorithm for the assortment optimization problem for general k. Through a series of computational experiments, we show that the proposed algorithm is easy to implement, efficient to run, and performs significantly better than its theoretical guarantee.

Third, we introduce the sequential flips nonparametric choice model. This model subsumes a classic linear utility model used when products are vertically

differentiated and a customer's utility of a product is a random linear function of the product's price and quality. By framing this model as a nonparametric choice model, we develop an efficient dynamic program to solve the corresponding assortment optimization problem. Further, this algorithm can be extended to the joint assortment and pricing optimization problem.

After introducing and studying all of these models, we lastly present a general approximation algorithm for the space constrained assortment optimization problem that applies to two of the models above. In this setting, the retailer wants to choose a subset of products to offer to maximize expected revenue but is restricted by a knapsack constraint on the total size of the offered assortment. This constraint might occur in online settings in which the retailer wants to choose which products to display on the first page of search results or in a setting where there is a fixed physical space to display products.

BIOGRAPHICAL SKETCH

Alice Paul grew up in Madison, NJ. She received her Bachelor of Science in mathematics from Harvey Mudd College.

ACKNOWLEDGEMENTS

I am grateful to my thesis adviser Professor David P. Williamson for his mentorship during my time at Cornell. His generous guidance on research and my career, support through the struggles of research, and deep subject knowledge have helped me grow as a researcher and person. I also appreciate the opportunities and example he has given me to develop as a teacher.

I would also like to thank my committee members Professor Huseyin Topaloglu and Professor David B. Shmoys for their help and contributions and give a special thanks to my co-author Professor Jake Feldman whom I have really enjoyed working with. Further, I would like to acknowledge the Department of Defense for the NDSEG fellowship support.

Last, I would like to thank my husband, my family, and Boo for all their love and support.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

CHAPTER 1

INTRODUCTION

Accurately modeling customer behavior is important for any retailer as this model will determine decisions about inventory, pricing, promotions, and more. One important factor for any choice model to incorporate is the dependence between which products a customer sees and their ultimate purchase. In particular, a customer may be willing to substitute between different products depending on availability. Customer choice models provide a way to model this interaction; a customer choice model maps any assortment of available products to the probabilities the products in the assortment are purchased. Through these models, we can also capture how a product's features affect its attractiveness and hence its probability of being purchased. A common feature that is considered is price, since it influences both demand and profit margins. Further, since the prices of each product can be varied after the products have been produced, it is a straightforward lever for the retailer to pull in an effort to optimize profits. A variety of customer choice models exist, each capturing the effects of substitution and price sensitivities differently.

One customer choice model to gain attention recently is the full nonparametric ranking-based choice model dating back to Mahajan and van Ryzin [44], [45]. In the full nonparametric ranking-based choice model, each customer class is distinguished by an arrival probability and a unique ranking on a subset of products referred to as a preference list. When presented with an assortment of products, a customer will purchase the highest ranking offered product in her preference list, and if there is no offered product in her preference list, then she leaves without making a purchase. As stated, this model places no restrictions on the set of potential preference lists, and hence, the number of customer types grows exponentially

in the number of products.

In a more complicated setting, customer classes also have budgets, and will purchase the highest ranking product in their preference list that is priced within their respective budget. We note that in modeling price sensitivity in this manner, we assume that prices only play a role in determining the consideration set of each customer. From a random utility maximization perspective, this amounts to the assumption that prices only influence the utility that a customer associates with each product in a binary manner: if a product is priced above the budget of a given customer, then this customer will associate a utility of zero with this product. Otherwise the associated utility can be viewed as a function of the other features of the product and is not influenced by price. This modeling assumption is validated by the work of Gilbride and Allenby [26], who study a two-stage choice model in which consumers first form consideration sets based on screening rules for the attributes of products, and then proceed to purchase the product with the highest utility within their consideration set. The authors find that choice models built on the groundwork of conjunctive screening rules, under which a product makes it into a consumer's consideration set only if it is found acceptable with regards to all relevant attributes, fit the data the best. The idea of a budget threshold is singled-out as one such attribute that could form the basis of a conjunctive screening rule. Further, using survey data on camera purchases, they find that price and body style play an important role in determining the consideration set, but not in the final choice from among the offered products. One can imagine that this extends to other settings in which customers purchasing a mid-range item have a threshold discount at which they will substitute to a higher-end product.

The clear benefit of the full nonparametric choice model is that by fitting this

model the retailer gains direct insight into purchasing patterns. Further, this model has the ability to generalize any random utility choice model by setting the associated probability of a preference list with the probability that the products' random utilities take on this ordering; however this modeling flexibility comes at a cost. Due to the potentially large number of customer types, it can be difficult to estimate the underlying arrival probabilities of each customer type and the accompanying revenue management problems are intractable. Specifically, under this model there is no efficient algorithm to determine the assortment that maximizes a retailer's expected revenue, a fundamental problem. In this thesis, we will consider restrictions on the nonparametric model that yield tractable estimation and optimization. An ideal choice model is one which is simple to describe, easy to estimate, and whose corresponding revenue management problems admit tractable and profitable solutions. The models we introduce will exhibit all these properties when the revenue management optimization problems we consider fall into two main categories: assortment and pricing problems.

The first model, which we present in Chapter 3, is the nonparametric tree choice model and is a specific case of the full nonparametric model. In this setting, we restrict the set of possible preference lists to be paths in an underlying tree. To be more precise, given an undirected tree in which each node corresponds to a unique product, the set of possible customer types is characterized by a set of paths in the tree. We restrict these paths to be linear in the sense that they must either move progressively towards or away from the root node. We fully formalize the notion of a linear path as well as the nonparametric tree model as a whole in Chapter 3. We note that it is best to use the nonparametric tree choice model in settings in which customers have monotonic preferences for features in a product line, and the retailer would like to understand how customers trade

off between various combinations of these features when making a purchase. For example, hotel purchases focus on features including price, discounts, bed size, square footage, the presence or absence of beautiful views, etc. The structure of the tree provides insights into how customers value these features when deciding which hotel room to purchase. In contrast, the nonparametric tree model may not be appropriate in a setting in which product features are difficult to compare (e.g. horizontally differentiated features).

The nonparametric tree model that we present should be viewed as a generalization of the intree and outtree models introduced in Honhon, Jonnalaegedda, and Pan [31]. These models are similar to ours in that customer classes correspond to paths in an underlying tree. However, both the intree and outtree models put extra restrictions on the structure of the paths associated with permissible customer classes, which ultimately leads to significant shortcoming in the practicality of the model. The intree model is appropriate when customers substitute from specific, specialized products to more general products. General products that are designed to appeal to a wide range of customers would be located towards the root of the tree, with the root being the most general. Products targeted to specific customer segments would be located towards the leaves of the tree, with the leaves being the most targeted products. The critical limitation of the intree model is that all preference lists include the root product, implying in our example, that all customers are willing to substitute to the most general product type. First, this means that if the product corresponding to the root node is made available for purchase, then every arriving customer will make a purchase. In settings in which there are high numbers of no-purchase events, such as e-commerce, it is likely that the intree model will fail to explain large portions of the sales data. Further, in assuming that all customer types substitute down to the root, the intree model is

unable to capture any differentiation in pickiness within the customer population. In Honhon et al. [31], the motivating example for the intree shows how it can be used to model customers purchasing various shampoos. The all-purpose shampoo is placed at the root, while shampoos targeted at very specific hair types are located at the leaves. The intree model assumes that all customers are willing to substitute to the all-purpose shampoo if their preferred, more targeted product is unavailable. In contrast, since the general nonparametric tree choice model allows for preference lists that can end anywhere, it allows us to capture the scenario in which a customer leaves the store without making a purchase if she cannot find her desired targeted product.

In the outtree model, all customer classes are associated with paths that begin at the root node and terminate at an interior or leaf node. The outtree model is appropriate when customers substitute from products with many features to products with less robust feature sets. This is the case, for example, in a product line that is targeted to a wide range of consumer budgets, with more expensive products having richer feature sets. Expensive products with many features would be located towards the root of the tree, with the most feature rich product at the root node. Less expensive products with less rich feature sets are located toward the leaves of the tree, with the least feature rich located at the leaves. The outtree model has similar limitations to the intree model: all preference lists include the root product as the highest ranked product, implying that all customers will purchase this product if it is offered. For customers who are budget conscious, this is clearly an unrealistic assumption. Again, since paths associated with preference lists can start at any node in the general nonparametric tree choice model, our model captures the various budgets associated with different segments of the customer population.

For the nonparametric tree model to be practically useful, it is also essential that its parameters can be estimated efficiently from sales data and that we can solve the common revenue management problems that arise in a tractable fashion. In Chapter 3, we show how to accomplish both of these tasks. In particular, we are able to show how to generate the tree structure from historical sales data rather than having the tree specified in advance, as in Honhon et al. [31]. This is an important differentiation since the structure, for example in the hotel setting above, may not always be clear or well-defined. Herein lies the importance of our estimation procedure, which discovers this structure for us. The combination of efficient estimation procedures for the model and tractable algorithms for the optimization problems allows the nonparametric tree model to form a practical basis of revenue management systems.

The second restriction on the full nonparametric model, which we study in Chapter 4, captures the setting in which customers are only willing to substitute a limited number of times. As before, customers purchase the highest ranking product available in their preference list. Since we hope to capture limited substitution, we only consider customers whose preference lists are of length at most k. We refer to this choice model as the k-product nonparametric choice model. There are numerous papers in the marketing and revenue management literature that provide evidence for the existence of limited substitution. In most of the past literature, customers that substitute a limited number of times have generally been referred to as customers that have small consideration sets, in which the consideration set of a customer is simply the set of products they would ever consider purchasing. Lapersonne, Laurent, and Le Goff [39] study consumers considering an automobile purchase and find that a large percentage of consumers will only consider the brand of their previous car. This sort of consumer behavior is also de-

scribed in Hauser, Ding, and Gaskin [28], in which the authors emphasize that for frequently purchased products, consumers often only consider a handful of brands. Further, in Jeuland [36], the author studies a purchasing bias which is referred to as short term brand loyalty or inertia in choice, in which consumers continue to buy the products or brands they have previously purchased. Finally, Crompton and Ankomah [11] study how travelers pick vacation destinations. They claim that in this setting, "consideration sets have size at most four". Additionally, with the growth of online retailers such as Amazon who offer millions of products, customers are generally able to get their first choice product online and hence it is hard to imagine that many consumers exhibit extreme forms of substitution behavior.

Last, in Chapter 5, we study the setting in which products are vertically differentiated. The model presented in this chapter is an extension of the classic linear utility model first seen in Mussa and Rosen [47] in which the random utility of each product is a linear function of the product's price and quality. Here, customers purchase the product with highest non-zero utility. We translate this linear model to a nonparametric choice model through the sequential flips nonparametric choice model. We show that when this classical model for vertically differentiated products is viewed from this new vantage point, then the optimization problems become more tractable. In the sequential flips nonparametric choice model, the set of preference lists is formed by starting with an initial preference list and iteratively flipping sequential products in the list to generate new lists. This restriction on the generation of lists is enough to yield tractable assortment and pricing optimization problems.

As mentioned above, the sequential flips nonparametric choice model is appropriate when products are vertically differentiated. In such a setting, the two

features that a consumer focuses on are the price and quality of each product. Quality should be viewed as an aggregated score based on other features not including price. For example, with regards to phones, these features could include screen size and memory. When a set of products can be ordered based on an objective quality measure, it is said that these products are vertically differentiated. There are a variety of industries in which products are differentiated in such a way. For example, in the airline industry there is a clear-cut ordering of the quality of the potential tickets that travelers can buy: first class is preferred to business class which is preferred to coach.

Overall, the three choice models we present cover many practical revenue management settings and allow us to develop tractable estimation and optimization procedures. Further, each setting presents unique and interesting challenges requiring us to approach the problems using new techniques.

1.1 Contributions

We consider a variety of fundamental revenue management problems under the three models presented above. These problems fall into three main categories: estimation, assortment optimization, and pricing. Below we summarize our results for our models.

In Chapter 3, we consider assortment optimization problems under the nonparametric tree choice model. In the assortment optimization problem, the retailer is presented with a collection of products from which she must choose an assortment of products to offer to customers so as to maximize expected revenue. In this case, we show that the assortment problem under the nonparametric tree choice model

can be solved with a dynamic program. This dynamic program has a small state space and, consequently, leads to efficient optimization. The key insight that we make in the dynamic program boils down to the idea that the purchase probability of any item within an arbitrary assortment can be computed recursively with only knowledge of each product's closest offered predecessor in the tree. The dynamic program for the pure assortment problem can be extended to settings in which the retailer has additional cost considerations. We consider scenarios in which there are fixed costs to include products in the offered assortment and penalties when a customer is forced to substitute to a less preferred product. Substitution penalties model a loss of customer good will, a common consideration for retailers. Second, we extend the dynamic program to the cardinality constrained assortment optimization problem. In this problem the available products are grouped into categories and the retailer can offer a limited number of products from each category. In the simplest case, all products are in a single category and the retailer is constrained to have an assortment of limited size. We show that our dynamic program can be extended to solve the cardinality constrained problem. Last, we consider the case when the retailer is allowed to break the tree structure slightly by considering general graphs on the set of products with bounded tree width. We show that our dynamic program can also be extended to this setting given constant tree width.

The second problem that we consider has come to be known as the joint assortment and pricing problem. In this problem, the retailer must choose an assortment of products to offer to customers as well as the prices for these offered products with the goal of maximizing the expected revenue from each arriving customer. In order to capture each consumer's sensitivity to price, we assume that each customer class is distinguished by a budget in addition to an arrival probability and

preference list. Arriving customers will purchase the highest ranking product in their respective preference list that is priced within their budget. It is not difficult to see that this problem generalizes the pure assortment problem and hence it is no surprise that the additional pricing element renders this problem more difficult. As a result, we place additional restrictions on the set of potential customer classes under the nonparametric tree choice model. We first assume that all preference lists are derived from a line graph, that is a tree consisting of a single path from the root to a leaf node. We call this model the interval model since all preference lists will be of the form $[i, i+1, \ldots, j]$. When prices are exogenous, this model reduces to the one-way substitution model of Honhon, Jonnalagedda, and Pan [31]. It is important to note that endogenizing prices in the manner we do renders the techniques and ideas presented for the one-way substitution model in Honhon et al. [31] irrelevant to our setting. As such, we provide the first polynomial-time algorithm for this problem when there are no restrictions on the set of prices that the retailer can charge. Second, we consider the joint assortment and pricing problem for the nonparametric tree choice model when prices are restricted to be quality consistent. In this setting, our dynamic programming ideas for the pure assortment optimization problem extend to incorporate prices.

In addition to considering the above assortment and pricing problems, we also provide evidence for the practical importance of the nonparametric tree choice model. To do so, we provide a heuristic for building the tree of products from historical sales data. We run two sets of experiments; the first set uses synthetic sales data generated from a known ground choice model and the second uses the real hotel booking data provided in Bodea, Ferguson, and Garrow [8]. We show that the fitted nonparametric tree choice models derived from the estimated tree structures capture customer behavior better than the well-known multinomial logit (MNL)

choice model for both sets of experiments. For the experiments based on synthetic sales data, we also find the optimal assortments under the fitted nonparametric tree model and the fitted MNL model respectively. Then, we check the performance of these recommended assortments under the ground truth choice model, which we used to generate the data. We find that the assortments recommended by the nonparametric tree model often outperform those recommended by the MNL model by over 10%. These results give evidence against the use of revenue ordered assortments, which are well known to be optimal under the MNL choice model and often employed because of their intuitive nature. For the hotel dataset, we do not know the true ground choice model so we test the models based on the metric of log-likelihood. We find that the nonparametric tree model outperforms the MNL model by an average of 3% across the two hotel datasets.

Next, in Chapter 4 we study the assortment optimization problem when customers make purchases according to the k-product nonparametric choice model. We prove that this problem is NP-hard even when $k = 2$ and when the set of preference lists is derived from a single ordering of the products, meaning that the products can be indexed such that a product with a lower index is never ranked below a product with a higher index in any preference list. We show this result via a reduction from the vertex cover problem on cubic graphs. Motivated by this hardness result, we begin by studying the assortment optimization problem under the 2-product nonparametric choice model. This setting captures the scenario in which customers substitute at most once if their most preferred product is not available for purchase. This problem exhibits more structure than the assortment optimization problem under the general k-product nonparametric choice model. In particular, we are able to reduce the assortment optimization problem to a maximum directed cut problem. This reduction yields a 1.14-approximation algorithm

that rounds the optimal solution to a semidefinite programming formulation of the problem. Further, we show that the revenue function under the 2-product nonparametric model is submodular. Exploiting this structure, we expand upon the ideas in Buchbinder et al. [10] and Poloczek et al. [50] and present a linear time 2-approximation algorithm.

Since these results cannot extend to longer preference lists, we use a different approach when customers substitute more than once. The assortment optimization problem under the k-product nonparametric choice model can be formulated as an integer program with binary decision variables denoting whether or not each product is offered. The algorithm that we present considers rounding the optimal decision variables for the linear programming relaxation of this integer program. Specifically, we offer each product i independently with a probability that is increasing in its corresponding optimal decision variable value and decreasing in k. The performance of our algorithm is worst in settings in which the optimal assortment garners a large fraction of its revenue from lower ranked products. A simple worst-case analysis shows that our randomized rounding algorithm is a 3.375-approximation algorithm for the assortment optimization problem under the 3-product nonparametric choice model and a 4.741-approximation algorithm under the 4-product nonparametric choice model. Aouad et al. [3] propose a simpler randomized approach, which offers each product with probability $1/k$. They prove a $(e \cdot k)$-approximation ratio for this algorithm. Our algorithm improves upon this theoretical guarantee for all k. For the case when $k = 2$, we show how to modify this algorithm to extend to the cardinality constrained version of the problem, in which the retailer can only offer a limited number of products. In this setting, the typical derandomization procedure cannot be applied due to the constraint limiting the total number of products that can be offered. Instead, we employ a

technique known as pipage rounding that carefully rounds products in pairs.

In order to test the efficacy of our proposed linear programming based algorithm, we run a series of computational experiments on a large collection of instances of the assortment problem under the k-product nonparametric choice model for $k = 3, 4$. Our proposed algorithm performs quite well in practice, far exceeding the worst case theoretical performance guarantee. Further, we compare the performance of our algorithm against the randomized approach of Aouad et al. [3]. We show that our algorithm outperforms this other approach both in terms of average and worst case performance over all test cases. We conclude Chapter 4 with another set of computational experiments that studies the marginal benefit of fitting increasingly complex nonparametric choice models. We generate sushi sales data using the 5000 rankings of ten different sushi rolls provided in Kamishima [37]. We then fit k-product nonparametric choice models for $k \in \{1, 2, 3, 4, 5\}$ to study the trade-off between the added accuracy that results from increasing k and the computational effort required to fit these richer nonparametric choice models. We find that the marginal gains in accuracy from increasing k beyond 4 are minimal in the setting we study, providing further support for considering choice models with limited substitution behavior.

In Chapter 5, we first introduce the classic choice model for vertically-differentiated products, which assumes that the utility that a customer associates with each product is a linear function of the quality and price of the product. Specifically, the utility that an arriving customer associates with product i is given by $U_i = \theta q_i - p_i$. In this expression, q_i and p_i are the quality and price of product i, respectively, and θ is a random variable with distribution $F(\theta)$ that captures how the arriving customer values quality over price when making a purchasing

decision. We show that the sequential flips nonparametric choice model subsumes this model. We then provide a simple dynamic programming approach to solve the corresponding assortment optimization problem, which essentially boils down the insights made in Pan and Honhon [49], who consider the same assortment problem. The dynamic program that we develop shows that we can make optimal offer decisions for each product by processing the customer classes in a special order. If n is the number of products available to the retailer, then we show that we can compute the optimal assortment in time $O(n^2)$. This improves upon the previous best known approach by a factor of $O(n)$.

The dynamic program presented has several extensions. First, the dynamic program extends to the setting in which there are fixed costs to include products in the offered assortment and penalties when a customer is forced to substitute to a less preferred product. Second, we extend the dynamic program to the cardinality constrained assortment optimization problem, in which the retailer can offer a limited number of products. Last and most surprising, we show how this dynamic program can be extended to the discrete pricing and assortment setting under certain restrictions.

Last, in Chapter 6, we consider the space constrained version of the assortment problem in which the retailer wants to find the revenue maximizing assortment to offer subject to a knapsack constraint on the total space consumption of the offered products. First, we show that the space constrained assortment problem under any of the three models we have presented is NP-Hard via a reduction from the knapsack problem. Motivated by this result, we develop a two-step procedure which leads to a 3-approximation algorithm for this problem. The first step of this procedure can be generally applied to any random utility customer choice

model and hence our approach has the potential to be useful beyond the scope of nonparametric choice models.

1.2 Literature Review

The nonparametric choice model was introduced to avoid some of the shortcomings of traditional parametric customer choice models. The most classic customer choice model is the multinomial logit (MNL) choice model. This model is easy to estimate from historical sales data and yields tractable optimization problems (see [56] and discussion below). However, the simplicity of the model may be unrealistic in practice since it assumes that comparisons between two products are unaffected by other products offered and can fail to capture simple substitution behavior. More complicated parametric models such as the nested logit model and mixed MNL model help to alleviate these limitations. However, under these more complicated models, estimating the parameters and overall structure becomes more difficult and the corresponding optimization problems become intractable except under certain variants.

To our knowledge, the nonparametric choice model first appeared in Mahajan and van Ryzin [44] and Mahajan and van Ryzin [45]. However, the model was formally proposed by Rusmevichientong, Van Roy, and Glynn [55] as an alternative customer choice model that can capture any general random utility model without imposing the more complicated structure of parametric models. The authors motivate this model in the context of pricing automobiles. Further, Farias, Jagabathula, and Shah [19] and van Ryzin and Vulcano [58] develop efficient estimation procedures for the general nonparametric choice model. Both papers present com-

putational experiments for these methods on synthetic and real historical purchase datasets.

There are a several papers that have considered the assortment optimization problem under the nonparametric choice model. The first paper to do so and the work that is most closely related to our work on the nonparametric tree choice model is Honhon, Jonnalagedda, and Pan [31], which considers the assortment optimization problem restricted to intrees and outtrees. As mentioned previously, both of these models have restrictions on which preference lists can be associated with customer classes. We extend the results of this paper by lifting many of these restrictions and working in a more general setting. In Honhon et al. [31], the authors also introduce operational considerations, such as fixed costs for introducing products or penalties when a customer substitutes to a less preferred product. We extend these results by introducing the cardinality constrained version of the assortment problem.

Two other papers that are closely related to our work are Aouad, Farias, and Levi [3] and Aouad, Farias, and Levi [2]. The former proves various hardness results related to the assortment problem under the nonparametric choice model. The latter considers the assortment optimization problem under the nonparametric choice model when customer preference lists are associated with structured set systems defined over a single overarching ordering of the products; one such structured set system is a laminar family. The general algorithm provided in this paper can be used to solve the outtree case described in Honhon et al. [31], but it does not generalize to the more complex intree case or the other settings we study.

Our work on the joint assortment and pricing problem under the interval model most closely resembles the work of Jagabathula and Rusmevichientong [35], who

consider the same joint assortment and pricing problem under the most general form of the nonparametric choice model. Under this more general form, the joint assortment and pricing problem is NP-Hard. Motivated by this result, the authors present a polynomial-time approximation scheme (PTAS) whose runtime scales exponentially in a parameter they call d, which essentially represents how much any feasible pricing scheme is allowed to break a quality consistent structure. Their approach relies on a fairly intricate dynamic program which places a carefully chosen grid on the set of prices that the retailer can charge. In contrast, while we consider a less general choice model, the algorithms that we provide are optimal and their runtime is polynomial in all input parameters. It is also important to note that Jagabathula and Rusmevichientong [35] also show how to estimate such a choice model from historical sales data, and thus their approach could also be applied to derive estimates for the parameters in our setting.

There are a few earlier works that consider the joint assortment and pricing problem under the nonparametric choice model. Aggarwal, Feder, and Motwani [1] are the first to develop algorithms with provable performance guarantees for variations of the joint assortment and pricing problem. Most notably, when the prices are constrained by a price ladder, the authors are able to develop a PTAS. This price ladder is essentially akin to restricting the prices to be quality consistent. Rusmevichientong, Van Roy, and Glynn [55] show that the joint assortment and pricing problem under the most general form of the budgeted nonparametric choice model is NP-Complete in the strong sense. Motivated by this result, they also restrict the set of feasible prices to a price ladder. With this simplification of the pricing structure, the authors develop various heuristics which they show work well in practice.

Work on the assortment optimization problem under the k-product nonparametric choice model is somewhat limited. Most notably, Berstimas and Mišic [6] present an integer program for this problem. Our proposed algorithm is based on the linear programming relaxation of this formulation. In addition to their various hardness results, Aouad, Farias, and Levi [3] show that randomly offering each product with probability $1/k$ produces a $(e \cdot k)$-approximation algorithm. To our knowledge, this is the best previously known guarantee for this problem. The 2-product nonparametric choice model also closely resembles the substitution model outlined in Kok and Fisher [38]. In this setting, the single substitution event that occurs when a customer's first choice product is unavailable unfolds in two steps. First, the customer decides whether to consider a second product or to leave the store. If she chooses to consider a second product, she then substitutes into one of the other products with a probability that is dependent on the customer's favorite product. The authors give heuristics for the associated assortment optimization problems.

The work we do regarding the sequential flips nonparametric choice model is most closely related to the work in Pan and Honhon [49]. The authors of this paper study both pricing and assortment problems under the linear utility model described in the previous section in which the random utility that each arriving customer associates with each product is given by $U_i = \theta q_i - p_i$. We give a thorough description of this model, which we call the linear utility choice model, in Chapter 5. The authors of this paper show that the pricing and assortment problems can be reduced to a tractable shortest path problem whose size scales nicely with the number of products. Our dynamic programming approach for the assortment optimization problem under the sequential flips nonparametric choice model builds upon this approach by viewing choice through the lens of the nonparamet-

ric choice model. Further, we extend our dynamic program to the *discrete* pricing and assortment problem in which prices must be chosen from a fixed list (e.g. five dollar increments). In this discrete setting, the methods presented by Pan and Honhon [49] do not apply.

As mentioned above, assortment optimization has also been thoroughly studied under other choice models. The most classic model is the multinomial logit (MNL) choice model. Talluri and van Ryzin [56] solve the assortment optimization problem under this model. Extending this result, Rusmevichientong, Shen, and Shmoys [52], Wang [60], Davis, Gallego, and Topaloglu [12], and Wang [61] study various versions of the constrained assortment problem under the MNL model. Additionally, Wang [60] solves the joint assortment and pricing problem under the MNL model. When parameter values are uncertain, Rusmevichientong and Topaloglu [54] study the corresponding robust assortment problem.

Work on the MNL choice model has also been extended to more complicated optimization problems. Specifically, Davis, Topaloglu, and Williamson [14], Aouad and Segev [5], and Gallego et al. [23] consider the problem of optimally displaying products when customers make purchases under the MNL model but might not view all products. Further, Aouad, Levi, and Segev [4] study the dynamic assortment problem in which stockouts may affect which products are available.

The MNL model can be enhanced through the more complicated mixed MNL model, which allows a distribution of multinomial logit models, and the nested logit model, which incorporates correlation between products. Méndez-Díaz et al. [46], Désir and Goyal [16], and Rusmevichientong, Shmoys, and Tong [53] focus on assortment optimization problems when customer choices are governed by the mixed MNL model. Further, Feldman and Topaloglu [21] give upper bounds on optimal

solutions for this problem. For the assortment problem under the nested logit model, Li and Rusmevichientong [41], Davis, Gallego, and Topaloglu [13], and Li and Huh [42] develop efficient solution methods. Extending these results, Gallego and Topaloglu [24] and Feldman and Topaloglu [22] consider the space and cardinality constrained versions of the assortment problem under the nested logit model. Lastly, Gallego and Wang [25], Rayfield, Rusmevichientong, and Topaloglu [51], and Davis, Topaloglu, and Williamson [15] study the joint assortment and pricing problem under the nested logit model.

More recently, Blanchet, Gallego, and Goyal [7] introduce the Markov chain choice model and shows that it subsumes the MNL model in addition to approximating other well-known choice models quite accurately. Désir et al. [18] extend this result to the cardinality constrained version of the problem. Further, Hosseinalifam, Marcotte, and Savard [32] show that the Markov chain choice model subsumes any nonparametric choice model in which the preference lists are nested, meaning they take the form $[1, 2, \ldots j]$. However any deviation from this nested structure on the preference lists breaks the validity of the reduction presented by Hosseinalifam, Marcotte, and Savard [32]. Further, it is also important to note that there is currently no efficient way to estimate the arrival and transition probabilities of the Markov chain choice model.

Finally, Jagabathula [34] considers the efficacy of greedy algorithms under a general random utility choice model when the retailer is given access to an oracle that can evaluate the revenue of any given assortment, and Désir et al. [17] consider assortment optimization problems under the Mallows model, which has similarities to the nonparametric choice model.

CHAPTER 2

GENERAL NOTATION

In this chapter, we introduce some general notation we will need for the customer choice models and optimization problems we study. For context, we first define the assortment optimization problem in its most general form and introduce the random utility choice model and its subvariant, the multinomial logit choice model. Next, we introduce the nonparametric choice model, which will form the basis for the models studied in this thesis, and present the remaining optimization problems using this framework.

We assume that a retailer has access to a set of n potential products given by $N = \{1, \dots, n\}$, each associated with a revenue $r_j \geq 0$. We refer to a product as offered if it is available for purchase and not offered if not. Given a customer choice model, the first problem that is often considered is the assortment optimization problem. In the **assortment optimization problem**, the retailer wants to decide which subset $S \subseteq N$ of products to offer to maximize the expected revenue from an arriving customer. In particular, given a subset of offered products $S \subseteq N$, we let $\Pr_j(S)$ be the probability a customer will purchase product j when the set S is offered. We can also encode the offered assortment through the vector $x \in \{0,1\}^n$, where $x_j = 1$ indicates that product j is offered and $x_j = 0$ indicates that it is not. We slightly abuse notation and substitute between S and x depending on the setting (i.e. using $\Pr_j(x)$). These purchase probabilities are determined by the corresponding customer choice model.

In the **random utility choice model**, every product i is associated with a known feature vector z_i. Given these vectors, the random utility of product i is

$$U_i = \beta^T \cdot z_i + \epsilon_i,$$

where β is a known vector and ϵ_i is an unobserved random variable. Further, the utility for the customer to leave the system without making a purchase is $U_0 = \epsilon_0$, also randomly distributed. Under this choice model, an arriving customer will purchase the product with highest utility above U_0 that is offered. In other words, the probability i is purchased when $S \subseteq N$ is offered is

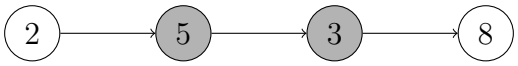
$$\Pr{}_i(S) = \begin{cases} \Pr[U_i = \max_{j \in S \cup \{0\}} U_j] & \text{if } i \in S, \\ 0 & \text{otherwise.} \end{cases}$$

The classic **multinomial logit model** (MNL) makes the added assumption that the ϵ_i variables are independent and drawn from a Gumbel distribution. In this case, the probability $i \in S$ is purchased simplifies to

$$\Pr{}_i(S) = \frac{\exp(\beta^T \cdot z_i)}{1 + \sum_{j \in S} \exp(\beta^T \cdot z_j)}.$$

The **nested logit model** expands upon this structure by allowing the ϵ_i variables to be correlated, and the **mixed multinomial logit model** incorporates a distribution over β.

In contrast, the nonparametric choice model removes the limitation of having to know the structure and distribution of the utilities. In the **general nonparametric choice model**, we are instead given a distribution over a set of preference lists given by a set of customer classes $\mathcal{G}$ (set $m = |\mathcal{G}|$). A customer in customer class $g \in \mathcal{G}$ arrives with probability λ_g and is associated with a preference list σ_g containing a subset of products. We assume without loss of generality that $\sum_{g \in \mathcal{G}} \lambda_g = 1$. For $i \in \sigma_g$, let $\sigma_g(i)$ give the index of product i in customer type g's preference list (with $\sigma_g(i) = \infty$ if $i \notin \sigma_g$). We use the convention that lower indexed products have a higher ranking. So a product that is first in a customer's preference list has the highest ranking. An arriving customer will purchase her highest ranked offered item (if any). In particular, if the retailer offers an assortment given by



$$\sigma_g(2) = 1 \quad \sigma_g(5) = 2 \quad \sigma_g(3) = 3 \quad \sigma_g(8) = 4$$

Figure 2.1: A customer of type g with preference list $[2, 5, 3, 8]$ will purchase product 5 when the assortment $S = \{3, 4, 5, 7\}$. Arrows represent possible substitutions.

$x \in \{0,1\}^n$, then a customer of type g will purchase product

$$\pi_g(x) := \begin{cases} \arg\min_{i \in \sigma_g, x_i = 1} \sigma_g(i) & \text{if } \sigma_g \cap \{i : x_i = 1\} \neq \emptyset, \\ 0 & \text{otherwise,} \end{cases}$$

where $\pi_g(x) = 0$ indicates that the customer leaves the system without making a purchase. We sometimes refer to this option as product 0 or the no-purchase option. See Figure 2.1.

Thus, given an assortment $x \in \{0,1\}^n$, the probability that a product j is purchased is $\Pr_j(x) = \sum_{g:\pi_g(x)=j} \lambda_g$ and the overall expected revenue is

$$\text{Rev}(x) = \sum_{j \in N} r_j \Pr_j(x).$$

Our objective in the assortment optimization problem is to find the assortment that maximizes this expected revenue. We denote this optimal revenue as

$$\text{OPT} = \max_{x \in \{0,1\}^n} \text{Rev}(x). \tag{2.1}$$

Aouad et al. [3] show that problem in Equation 2.1 is NP-Hard to approximate within a factor of $O(n^{1-\epsilon})$ for any $\epsilon > 0$. Further, the hardness result of [3] holds even when the preference lists for each customer type are constructed from a single overarching ordering, i.e. there exists an ordering $\prec$ on the products where $\sigma_g(i) \leq \sigma_g(j)$ implies $i \prec j$ for all $g \in \mathcal{G}$.

In certain settings the number of products to display to customers may be limited. For example, in online settings the retailer might want to choose which

products to display on the first page of results, and in physical settings there may be limited display space. In the **space constrained assortment optimization problem**, every product is also associated with a size $c_i \geq 0$ and the retailer wants to choose an assortment $S \subseteq N$ to maximize expected revenue with the added constraint that $\sum_{i \in S} c_i \leq C$, where C is a fixed constant. In the case that $c_i = 1$ for all $i \in N$ this is referred to as the **cardinality constrained assortment optimization problem**. As one can imagine, this added constraint can make the optimization problem much harder.

Further, we consider the more complicated **joint assortment and pricing problem**. In this problem, the retailer also has control over the prices of products. The retailer then must choose an assortment of products to offer to customers as well as the prices for these offered products again with the goal of maximizing the expected revenue from each arriving customer. In order to capture each consumer's sensitivity to price, we assume that each customer class also has a budget in addition to an arrival probability and preference list. Specifically, in this setting, each customer class $g \in \mathcal{G}$ is distinguished by an arrival probability λ_g, a preference list σ_g, and a budget b_g. Let $\{b_1, b_2, \ldots, b_d\}$ be the set of budgets for all customers in $\mathcal{G}$. We assume the budgets are indexed such that $b_i \leq b_{i+1}$ for all $i = 1, \ldots, d-1$. An arriving customer will purchase the highest ranking product that is priced within her respective budget, if any.

Suppose again we have products N. If a customer purchases product i priced at $p_i \in \mathcal{R}_+$, then the retailer makes a revenue of $p_i - k_i$, where k_i is the fixed unit cost of acquiring one unit of product i. We represent our pricing decisions using the vector $p \in \mathcal{R}_+^n$ and use the convention that setting $p_i = \infty$ is equivalent to not offering the product. If the retailer sets prices p, then a customer of type g will

purchase product

$$\pi_g(p) := \begin{cases} \text{argmin}_{l \in \sigma_g : p_l \leq b_g} \, l & \text{if } \sigma_g \cap \{i : p_i \leq b_g\} \neq \emptyset, \\ 0 & \text{otherwise.} \end{cases}$$

Therefore, the probability product i is purchased under prices p is $\Pr_i(p) = \sum_{g:\pi_g(p)=i} \lambda_g$. We can find the optimal assortment and prices to offer by solving the problem

$$\text{OPT}_p = \max_{p \in \mathcal{R}_+^n} \sum_{i \in N} (p_i - k_i) \Pr{}_i(p). \tag{2.2}$$

We can simplify the problem by noting that we can restrict the prices of p to the set $\{b_1, \ldots, b_d, b_{d+1}\}$, where $b_{d+1} = \infty$. The following Lemma shows this result.

Lemma 2.0.1. *There exists an optimal pricing scheme p^* to problem (2.2) where $p^* \in \{b_1, \ldots, b_d, \infty\}^n$.*

Proof. Assume that there exists $p_i^* \in p^*$ such that $b_j < p_i^* < b_{j+1}$ for some $j = 1, 2, \ldots, d$. Consider a new pricing scheme $\hat{p} = (p_1^*, \ldots, p_{i-1}^*, b_{j+1}, p_{i-1}^*, \ldots, p_n^*)$, which is simply the original pricing scheme except that we have increased the price of product i to b_{j+1}. We will show that this new pricing scheme achieves an expected revenue that is at least that of p^*. Consider a customer class $g \in \mathcal{G}$ that purchases product i under prices p^*. Then, $b_g \geq p_i^*$. However, this implies that $b_g \geq b_{j+1}$ since no customer budget points lie in between p_i^* and b_{j+1}. Therefore, this customer will continue to purchase i under prices $\hat{p}$ but for a higher price. Now consider a customer class $g \in \mathcal{G}$ that did not purchase product i under prices p^*. Since we raise the price of product i, this customer is not incentivized to switch to product i and will not change which product they purchase. Therefore, increasing the price of product i does not change which products are purchased but does

increase the overall revenue of the retailer. Thus, there exist optimal prices p^* such that $p^* \in \{b_1, \ldots, b_d, \infty\}^n$. $\square$

CHAPTER 3

NONPARAMETRIC TREE CHOICE MODEL

In this chapter, we present the nonparametric tree choice model as our first special case of the full nonparametric model. In this setting, we restrict the set of possible preference lists to be paths in an underlying tree. To be more precise, given an undirected tree in which each node corresponds to a unique product, the set of possible customer types is characterized by a set of paths in the tree. We restrict these paths to be linear in the sense that they must either move progressively towards or away from the root node. Under this model, the model parameters can be estimated efficiently from sales data and we can solve the common revenue management problems that arise in a tractable fashion. Further, we are able to show how to generate the tree structure from historical sales data rather than having the tree specified in advance, as in Honhon et al. [31]. This is an important differentiation since the structure may not always be clear or well-defined. The combination of efficient estimation procedures for the model and tractable algorithms for the optimization problems allows the tree model to form a practical basis of revenue management systems.

3.1 Model

Under the nonparametric tree choice model, customer classes are based on a rooted undirected tree structure $T = (N, E)$. The nodes in the tree represent all products that the retailer can potentially offer. Any customer class $g \in \mathcal{G}$ has a product preference list σ_g associated with a path in T; the ordering of the products in σ_g will correspond to the order products are visited in a path through T. We restrict

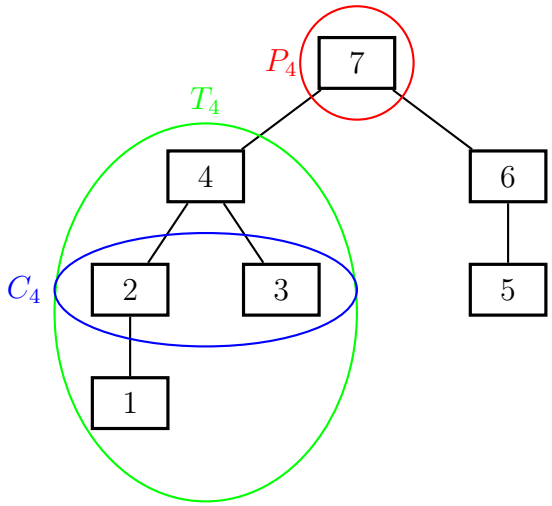


Figure 3.1: An example of a set of customer classes represented as a rooted binary tree. Possible customer preference lists are all linear paths including $[7, 4, 2, 1]$, $[3, 4]$, and $[5]$. The path $[1, 2, 4, 3]$ is not linear and would not correspond to a possible customer class.

our attention to *linear* paths, paths that visit at most one child of every node. See Figure 3.1. In an effort to solve the assortment problems for the most general form of the tree model, we allow the paths that we associate with preference lists to move towards or away from the root node. However, we emphasize that the main benefit of the nonparametric tree model is that paths associated with customer classes can start anywhere in the tree. When no confusion arises, we identify σ_g with the path in the tree and refer to the preference list as moving towards or away from the root. In what follows, we will assume the tree T is a binary tree. This assumption is without loss of generality; we can meet this requirement by adding at most n nodes that represent null products that provide no cost or benefit to the retailer.

3.2 Assortment Optimization

In this section, we provide a dynamic program for the assortment optimization problem under the nonparametric tree choice model. The structure of the underlying tree T will guide the steps of computation in our dynamic program. Before stating the dynamic program we first introduce additional notation and develop specific insights into the solution. Table 3.1 summarizes the various pieces of notation that we use.

Table 3.1: Tree notation.

T	$\triangleq$	Rooted tree structure (N, E)
T_i	$\triangleq$	Subtree rooted at node $i \in N$ containing i and all successors of i
C_i	$\triangleq$	Children of node $i \in N$ in the tree T
P_i	$\triangleq$	Parent of node $i \in N$ in the tree T
$\phi_i(S)$	$\triangleq$	For $S \subseteq N$ and $i \in S$, i's closest predecessor in S
$\delta_i(S)$	$\triangleq$	For $S \subseteq N$ and $i \in S$, the set of closest successors to i in S
$\Phi(i)$	$\triangleq$	All of i's predecessors in T in addition to the no-purchase option

Given a vertex i, we let C_i be the children of i in T. Further, we say that i is the parent of all $j \in C_i$ and define $P_j = i$. Note that for leaves of T, $C_i = \emptyset$. We will also be interested in complete subtrees of T. We let T_i be the subtree rooted at i containing i and all successors of i. When there is no confusion we will also use T_i to refer to the products represented by the nodes of the complete subtree. Without loss of generality, we can index the nodes such that the root node has index n and if $T_i \subset T_j$ then $j > i$. See Figure 3.1.

The tree T will be used to define *blocking* relationships among products. For a customer class g we say i blocks product j when S is offered if $\pi_g(S) = i$ and

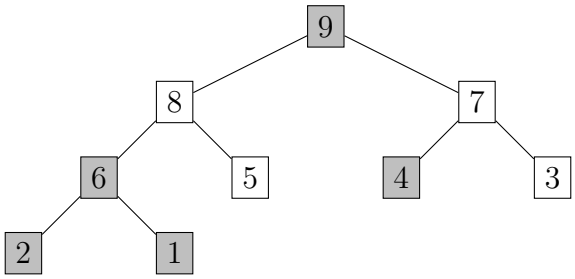


Figure 3.2: If we offer products $S = \{9, 4, 6, 2, 1\}$, then the closest offered predecessor of product 2 is $\phi_2(S) = 6$ and the closest offered successors of product 9 are $\delta_9(S) = \{6, 4\}$.

$\pi_g(S \setminus \{i\}) = j$. More generally, we say i blocks j when S is offered whenever there exists at least one class g for which this blocking relationship holds. Intuitively, i blocks j when the removal of i from the offer set induces a customer class to purchase product j. Since T defines the ordered lists for customer classes, these blocking relationships are tied to T. We define, for any pair of nodes, the degree to which they block each other. Specifically, we let

$$B_{i,j} = \sum_{g:\pi_g(\{i,j\})=i,\pi_g(\{j\})=j} \lambda_g.$$

Note that $B_{i,j}$ is not identical to $B_{j,i}$ since these two terms involve customer classes moving in opposing directions, which may have different associated probabilities. In addition to describing blocking in terms of probability, we will also describe blocking in terms of revenue. We let $r_j B_{i,j}$ be the revenue i blocks from j when $\{i, j\}$ is offered.

Given an assortment S and $i \in S$, we define $\phi_i(S)$ to be i's closest offered predecessor in S and $\delta_i(S) = \{j \in S | \phi_j(S) = i\}$ to be the set of closest offered successors to i in S. If no predecessor of i is offered in S, we let $\phi_i(S) = 0$. If no successors are offered, we let $\delta_i(S) = \emptyset$. Further, we use $\Phi(i)$ to represent all of i's predecessors in T in addition to product 0, the no-purchase option. See Figure 3.2.

If we offer assortment S and $i \in S$, any customer class g traveling away from the root that has i in its preference list does not end up purchasing i if and only if she purchases a predecessor j of i. Since all customer classes are linear, this customer class must also contain $\phi_i(S)$ in σ_g before i. Therefore, we know σ_g contains both $\phi_i(S)$ and i but ranks $\phi_i(S)$ above i. Similarly, a customer class traveling up the tree towards the root that has i in its preference list does not purchase i if and only if it purchases a successor of i. Since all customer classes are linear, σ_g must contain both a node $j \in \delta_i(S)$ and i but ranks j above i. These considerations allow us to rewrite the probability i is purchased when S is offered using our blocking notation:

$$\Pr_i(S) = \begin{cases} \Pr_i(\{i\}) - B_{\phi_i(S),i} - \sum_{j \in \delta_i(S)} B_{j,i} & i \in S \\ 0 & i \notin S \end{cases}. \tag{3.1}$$

This alternative expression is critical in our dynamic programming formulation. Note that this probability does not change if i's closest offered predecessor and closest offered successors remain the same. In essence, purchase probabilities related to i depend on local decisions.

Our dynamic program is based on maximizing *adjusted revenues* in complete subtrees of T. Intuitively, given T_i and a node $p \in \Phi(i)$, the adjusted revenue of an offer set $S_i \subseteq T_i$ is the revenue received from products in S_i when we offer $S_i \cup \{p\}$ minus the revenue S_i blocks from the product p. More precisely, given a subset $S_i \subseteq T_i$ and a closest offered predecessor p of i, we define the adjusted revenue of S_i to be

$$A(S_i, p) = \sum_{j \in S_i} r_j \Pr{}_j(S_i \cup \{p\}) - r_p \sum_{k \in \delta_p(S_i \cup \{p\})} B_{k,p}.$$

Note that this expression also holds when $p = 0$. The first term is the revenue received from products in S_i when the offer set is $S_i \cup p$. The second term accounts

for the revenue S_i blocks from p. The proposition below shows that the revenue of any assortment can be computed by summing adjusted revenues.

Proposition 3.2.1. *Consider any subset $S \subseteq N$ and node i with children l and r. Let $S_i = T_i \cap S$, $S_l = T_l \cap S$, and $S_r = T_r \cap S$. Lastly, let $p = \phi_i(S)$. Then,*

$$A(S_i, p) = \begin{cases} A(\{i\}, p) + A(S_l, i) + A(S_r, i) & i \in S, \\ A(S_l, p) + A(S_r, p) & i \notin S \end{cases}$$

In particular, this shows that

$$A(S, 0) = \sum_{i \in S} A(\{i\}, \phi_i(S)) = \text{Rev}(S).$$

Proof. First, suppose $i \in S$. Then, $\delta_p(S_i \cup \{p\}) = \{i\}$ since it is the only node in S_i for which p is the closest offered predecessor, i.e. $\phi_i(S_i \cup \{p\}) = p$. We have

$$\begin{aligned} & A(S_i, p) \\ &= \sum_{j \in S_i} r_j \Pr{}_j(S_i \cup \{p\}) - r_p \sum_{k \in \delta_p(S_i \cup \{p\})} B_{k,p} \\ &= \sum_{j \in S_i} r_j \left(\Pr{}_j(\{j\}) - B_{\phi_j(S_i \cup \{p\}), j} - \sum_{k \in \delta_j(S_i \cup \{p\})} B_{k,j} \right) - r_p B_{i,p} \\ &= r_i \Pr{}_i(\{i\}) - r_i B_{p,i} - r_p B_{i,p} + \sum_{j \in S_i \setminus \{i\}} r_j \Pr{}_j(S_i \cup \{p\}) - r_i \sum_{k \in \delta_i(S_i \cup \{p\})} B_{k,i} \\ &= r_i \Pr{}_i(\{i\}) - r_i B_{p,i} - r_p B_{i,p} + \sum_{j \in S_i \setminus \{i\}} r_j \Pr{}_j(S_i) - r_i \sum_{k \in \delta_i(S_i)} B_{k,i} \end{aligned}$$

where the second line follows from Equation 3.1 and the last line comes from the fact that for each $j \in S_i \setminus \{i\}$, the closest predecessor of j is not p (since i is offered) and so $\Pr_j(S_i \cup \{p\}) = \Pr_j(S_i)$.

We can simplify this expression further. The set S_i can be decomposed into $\{i\}$ and two additional sets: $S_l = S_i \cap T_l$ and $S_r = S_i \cap T_r$. Let $j \in S_i \setminus \{i\}$. Without loss

of generality, let $j \in T_l$. Then, $\phi_j(S_i) = \phi_j(S_l \cup \{i\})$ since its closest predecessor must be i or in the same subtree as j. This also shows $\delta_j(S_i) = \delta_j(S_l \cup \{i\})$ and

$$\Pr{}_j(S_i) = \Pr{}_j(S_l \cup \{i\}).$$

Lastly, we can easily see that

$$\delta_i(S_i) = \delta_i(S_l \cup \{i\}) \cup \delta_i(S_r \cup \{i\}).$$

Continuing from the above expression, these observations allow us to write

$$\begin{aligned}
A(S_i, p) &= r_i \Pr{}_i(\{i\}) - r_i B_{p,i} - r_p B_{i,p} \\
&\qquad + \sum_{j \in S_l} r_j \Pr{}_j(S_l \cup \{i\}) - r_i \sum_{k \in \delta_i(S_l \cup \{i\})} B_{k,i} \\
&\qquad + \sum_{j \in S_r} r_j \Pr{}_j(S_l \cup \{i\}) - r_i \sum_{k \in \delta_i(S_r \cup \{i\})} B_{k,i} \\
&= r_i \Pr{}_i(\{i\}) - r_i B_{p,i} - r_p B_{i,p} + A(S_l, i) + A(S_r, i) \\
&= r_i \Pr{}_i(\{i, p\}) - r_p B_{i,p} + A(S_l, i) + A(S_r, i) \\
&= A(\{i\}, p) + A(S_l, i) + A(S_r, i).
\end{aligned}$$

The last inequality follows by the definition of the adjusted revenue and noting that $\delta_p(\{i, p\}) = p$. By very similar analysis, when $i \notin S$ we get

$$A(S_i, p) = A(S_l, p) + A(S_r, p).$$

Therefore, by unraveling the recursion given in the statement of the proposition

we get that

$$\begin{aligned}
A(S,0) &= \sum_{i\in S} A(\{i\},\phi_i(S)) \\
&= \sum_{i\in S} r_i \Pr{}_i(\{i\}) - r_i B_{\phi_i(S),i} - r_{\phi_i(S)} B_{i,\phi_i(S)} \\
&= \sum_{i\in S} r_i \Pr{}_i(\{i\}) - r_i B_{\phi_i(S),i} - r_i \sum_{j\in\delta_i(S)} B_{j,i} \\
&= \sum_{i\in S} r_i \Pr{}_i(S) \\
&= \mathrm{Rev}(S),
\end{aligned}$$

where the second to last equality follows by Equation 3.1. □

By Proposition 3.2.1, we can rewrite the assortment optimization problem under the nonparametric tree choice model as

$$\mathrm{OPT} = \max_{S\subseteq N} A(S,0).$$

We now present our dynamic programming formulation. Each stage is a product i under consideration for inclusion in S and the one dimensional state space is a product p, possibly equal to 0, that is the closest offered predecessor of i in T. Our value function $V_i(p)$ is the maximum adjusted revenue that can be achieved from subsets of T_i when p is the closest offered predecessor of i.

$$V_i(p) = \max\{r_i \Pr{}_i(\{i\}) - r_i B_{p,i} - r_p B_{i,p} + \sum_{k\in C_i} V_k(i), \sum_{k\in C_i} V_k(p)\}. \tag{3.2}$$

For leaves of T, our base case, this simplifies to

$$V_i(p) = \max\{r_i \Pr{}_i(\{i\}) - r_p B_{i,p} - r_i B_{p,i}, 0\}.$$

Theorem 3.2.2.

$$V_i(p) = \max_{S_i\subseteq T_i}\{A(S_i,p)\}.$$

Proof. First, consider the base case. For the leaves of T, $V_i(p) = \max\{r_i \Pr_i(\{i\}) - r_p B_{i,p} - r_i B_{p,i}, 0\}$. This first term is equivalent to $A(\{i\}, p)$ since $r_i \Pr_i(\{i, p\}) = r_i \Pr_i(\{i\}) - r_p B_{i,p}$ and the second term is $A(\emptyset, p)$ so the claim holds.

Now consider a node i that is not a leaf and suppose that the claim holds for all successors of i. Let l and r be the left and right children of i, respectively. Let $S_i^* \subseteq T_i$ be a subset that maximizes $A(S_i, p)$. In the first case, suppose $i \in S_i^*$. Then, $\delta_p(S_i^* \cup \{p\}) = \{i\}$ since it is the only node in S_i^* for which p is the closest offered predecessor, i.e. $\phi_i(S_i^* \cup \{p\}) = p$. From Proposition 3.2.1, we have that

$$A(S_i^*, p) = r_i \Pr{}_i(\{i\}) - r_i B_{p,i} - r_p B_{i,p} + A(S_l^*, i) + A(S_r^*, i).$$

Noting that S_i^* is the maximizer of $V_i(p)$ and that $A(S_l^*, i)$ and $A(S_l^*, i)$ are completely independent since they do not share any successors in the tree, we see that we can express our optimization problem recursively:

$$\begin{aligned}\max_{S_i \subseteq T_i : i \in S_i} A(S_i, p) &= r_i \Pr{}_i(\{i\}) - r_i B_{p,i} - r_p B_{i,p} + \max_{S_l \subseteq T_l} A(S_l, i) + \max_{S_r \subseteq T_r} A(S_r, i) \\ &= r_i \Pr{}_i(\{i\}) - r_i B_{p,i} - r_p B_{i,p} + V_l(i) + V_r(i)\end{aligned}$$

where we have used the inductive hypothesis. By very similar analysis, when $i \notin S_i^*$ we get

$$\begin{aligned}\max_{S_i \subseteq T_i : i \notin S_i} A(S_i, p) &= \max_{S_l \subseteq T_l} A(S_l, p) + \max_{S_r \subseteq T_r} A(S_r, p) \\ &= V_l(p) + V_r(p).\end{aligned}$$

By combining these expressions we reach the desired claim

$$\begin{aligned}\max_{S_i \subseteq T_i} A(S_i, p) &= \max\{\max_{S_i \subseteq T_i : i \notin S_i} A(S_i, p), \max_{S_i \subseteq T_i : i \in S_i} A(S_i, p)\} \\ &= \max\{r_i \Pr{}_i(\{i\}) - r_i B_{p,i} - r_p B_{i,p} + \sum_{k \in C_i} V_k(i), \sum_{k \in C_i} V_k(p)\}.\end{aligned}$$

□

The special case of Theorem 3.2.2 when $i = n$ and $p = 0$ shows that our dynamic program computes the optimal solution to the assortment optimization problem.

We can now analyze the computational complexity of computing the necessary $V_i(j)$. Let $\mathcal{D}$ be the depth of T. The depth of a tree T is the length of the longest path from the root to one of the leaves of the tree. We pre-compute each $\Pr_i(\{i\})$ and $B_{i,j}$. The number of customer classes is $|\mathcal{G}| = O(n\mathcal{D})$ since linear paths in the tree are uniquely determined by a starting and ending point in the tree. Each $g \in \mathcal{G}$ contributes to at most $\mathcal{D}^2$ $B_{i,j}$ values since $|\sigma_g| \leq \mathcal{D}$ and we need to consider each pair of nodes in σ_g. Each customer class also contributes to at most $\mathcal{D}$ $\Pr_i(\{i\})$ values. Therefore, calculating the $\Pr_i(\{i\})$ and $B_{i,j}$ values has running time $O(n\mathcal{D}^3)$. After this pre-computation, each of the $O(n\mathcal{D})$ values $V_i(j)$ can be computed in constant time. This leads to an overall running time of $O(n\mathcal{D}^3)$. For a full binary tree, $\mathcal{D} = \log n$, leading to a running time of $O(n \log^3 n)$.

3.3 Extensions of the Dynamic Program

The dynamic program in Equation 3.2 can be easily extended to other settings including additional costs, space considerations, and paths that slightly break the tree structure.

3.3.1 Additional Costs

In this section we focus on two costs proposed in Honhon et al. [31]: a setup cost incurred when offering a product and a substitution penalty incurred when

a customer is forced to substitute to less desirable products. To model setup costs we introduce a constant fixed cost of k_i for offering product i, which could represent a setup or stocking cost. To model the substitution penalty we introduce a function $f(l)$ that represents the penalty incurred when a customer purchases their l^{th} most preferred product. Specifically, if a customer of type g purchases product i and $l = \sigma_g(i)$ then the retailer incurs a penalty of $f(l)$. In Honhon et al. [31], the authors assume that $f(\cdot)$ is linear and increasing and that $f(1) = 0$; we consider arbitrary functions. Below we present an extension of our dynamic program that includes these costs. This provides a polynomial time algorithm for assortment optimization under these cost considerations and is an improvement over the exponential time algorithm of Honhon et al. [31].

We let

$$P_i = \sum_{g \in G : i \in \sigma_g} \lambda_g f(\sigma_g(i)).$$

be the sum of penalties the retailer incurs if set $S = \{i\}$ is offered. When offering i prevents a customer from substituting further down in their list, it can potentially lower the total penalty. This inspires a notion of "blocking" similar to that which we introduced in the previous section. Given any pair of nodes, we let

$$Q_{i,j} = \sum_{g \in G : \pi_g(\{i,j\}) = i, \pi_g(\{j\}) = j} \lambda_g f(\sigma_g(j))$$

be the penalty i blocks from j.

We can now write the modified dynamic program:

$$V_i(p) = \max\{r_i \Pr{}_i(i) - r_i B_{p,i} - r_p B_{i,p} - k_i - P_i + Q_{i,p} + Q_{p,i} + \sum_{k \in C_i} V_k(i), \sum_{k \in C_i} V_k(p)\}. \tag{3.3}$$

For leaves of T, our base case, this simplifies to $V_i(p) = \max\{r_i \Pr_i(i) - r_i B_{p,i} - r_p B_{i,p} - k_i - P_i + Q_{i,p} + Q_{p,i}, 0\}$. The value functions in Equation 3.3 capture the

adjusted revenue for product i and all of its successors given that product p is the closest offered successor of i. Here, $r_i \Pr_i(i) - r_i B_{p,i} - k_i - P_i + Q_{p,i}$ is the revenue received from i, modified to include costs and penalties, when p is the closest offered predecessor and other products in T_i are not offered. The term $-r_p B_{i,p} + Q_{i,p}$ adjusts both the revenue and the penalty term for p since some customers may choose i instead.

The addition of $Q_{i,j}$ and P_i only introduce a constant multiplier to the running time of the dynamic program since $Q_{i,j}$ and P_i can be pre-computed simultaneously with $B_{i,j}$ and $\Pr_i(\{i\})$. As a consequence, the running time of this modified algorithm remains $O(n\mathcal{D}^3)$ where $\mathcal{D}$ is the depth of T.

3.3.2 Cardinality Constraints

Realistically, retailers are under many constraints and are not able to offer an arbitrary set of products to their customers. In the simplest case, a single shelf space constraint, each item consumes a single unit of capacity and the retailer has $C \leq n$ units of capacity on her shelves. This constraint is akin to a limit on the total number of products that the retailer can offer. Recall the cardinality constrained assortment optimization problem:

$$\max_{S:|S|\leq C} \text{Rev}(S). \tag{3.4}$$

Adding this cardinality constraint couples decisions across different branches of the tree, complicating the assortment problem at hand.

We present a dynamic programming approach to solve the capacitated version of the problem. We will have a two dimensional state space: for each product i

that is under consideration for inclusion in S we store p (possibly equal to 0) that is a predecessor of i in T and c that is the remaining number of products in T_i that we have left to offer. Our value function $V_i(p, c)$ will be the maximum adjusted revenue that can be achieved from subsets of T_i by offering at most c products when p is the closest offered predecessor of i.

$$V_i(p,c) = \max\{r_i \Pr{}_i(\{i\}) - r_i B_{p,i} - r_p B_{i,p} + \max_{c_l, c_r : c_l + c_r \leq c-1} V_l(i, c_l) + V_r(i, c_r), \tag{3.5}$$

$$\max_{c_l, c_r : c_l + c_r \leq c} V_l(p, c_l) + V_r(p, c_r)\}.$$

For leaves of T, our base case, this simplifies to

$$V_i(p,c) = \begin{cases} \max\{r_i \Pr_i(\{i\}) - r_i B_{p,i} - r_p B_{i,p}, 0\} & c > 0, \\ 0 & c = 0. \end{cases}$$

This inner maximization represents an optimal allocation of the remaining products to i's left and right children, which we represent as nodes l and r respectively.

The value functions for the constrained problem resemble those of the unconstrained problem given in Equation 3.2, although we add an additional element to the state space to ensure we output a feasible assortment. The pre-processing of the $B_{i,j}$ and $\Pr_i(\{i\})$ values remains identical and takes $O(n\mathcal{D}^3)$ time. However, adding the additional state increases the number of necessary $V_i(j, c)$ values to $O(n^2\mathcal{D})$ and the computation for each value to $O(n)$. Therefore, the overall runtime is $O(\max\{n^3\mathcal{D}, n\mathcal{D}^3\}) = O(n^3\mathcal{D})$ since $\mathcal{D} \leq n$.

3.3.3 Bounded Tree Width

Lastly, we can extend to a setting where we slightly break the tree structure. Suppose instead that the products can be represented as a rooted undirected tree $T = (\mathcal{X}, E)$ where the nodes $\mathcal{X} = \{X_1, \ldots, X_t\}$ of T are subsets of products such that

1. $|X_i| \leq w$ for all $X_i \in \mathcal{X}$,
2. $X_1 \cup X_2 \ldots \cup X_t = N$, and
3. if $j \in X_{i_1}$ and $j \in X_{i_2}$, then all nodes X_k of the tree on the path between X_{i_1} and X_{i_2} also contain j.

We say that T has tree width w.

Again suppose that customers correspond to linear paths in this tree. In this case, for every linear path $X_{i_1}, X_{i_2}, \ldots, X_{i_p}$ in T, a customer can take on any ranking on the products in $X_{i_1} \cup \ldots \cup X_{i_p}$ such that if $j \in X_{i_k}$ but is not in X_{i_l} where $k < l$ then j is ranked before all products in $X_{i_l} \setminus X_{i_k}$. We say that such a ranking *agrees* with the tree T. We can think of this representation of products as an extension of the nonparametric tree setting where some products might have the same features but might be different brands or colors. Here, it doesn't make sense to put a strict way that customers might consider these items but instead might consider all possible rankings among these products. This setting allows us to group those products into a single node.

We extend the nonparametric tree dynamic program to this case. We will use the same notation as before with T: C_i will denote the children nodes of X_i, P_i will be the parent node of X_i, and T_i will be the subtree rooted at X_i

containing all of X_i's successors. Given an assortment $S \subseteq N$, we define the closest offered predecessor of X_i, $\phi_i(S)$, to be j such that X_j is the closest predecessor of X_i in T from which we offered at least one product in S. Similarly, we define $\delta_i(S) = \{X_j \in T | X_j \cap S \neq \emptyset, \phi_j(S) = i\}$ to be the set of closest offered successors to X_i in S.

Then, if we offer the assortment S, for a product $j \in S$ in node X_i, the probability that j is purchased is

$$\Pr{}_j(S) = \Pr{}_j(X_i) - \sum_{k \in X_{\phi_i(S)} \setminus X_i} B_{k,j} - \sum_{l \in \delta_i(S)} \sum_{k \in X_l \setminus X_i} B_{k,j}.$$

This follows directly from the linearity of customer classes. We say $S_i \subseteq X_i$ is *consistent* with $S_j \subseteq X_j$ if they agree on all products in $X_i \cap X_j$.

This more complicated version of blocking leads to a slightly different definition of adjusted revenue. Intuitively, given T_i and an assortment S_p of $X_p \in \Phi(X_i)$, the adjusted revenue of an offer set $S_i \subseteq T_i$ is the revenue received from products in $S_i \setminus S_p$ when we offer $S_i \cup S_p$, minus the revenue $S_i \setminus S_p$ blocks from the products in $S_p \setminus S_i$. More precisely, given a subset $S_i \subseteq T_i$ and an assortment S_p of $X_p \in \Phi(i)$, we define the adjusted revenue of S_i to be

$$A(S_i, S_p) = \sum_{j \in S_i \setminus S_p} r_j \Pr{}_j(S_i \cup S_p) - \sum_{k \in S_p \setminus S_i} r_k \sum_{l \in S_i \setminus S_p} B_{l,k}.$$

With this new notation we can rewrite

$$\text{Rev}(S) = \sum_{i=1}^{t} A(S \cap X_i, S \cap X_{\phi_i(S)}).$$

As before, each stage in our dynamic program is a node X_i where we're deciding all products in X_i to offer and the w dimensional state space is the assortment S_p, possibly empty, offered from the closest predecessor to X_i in T. Our value function

$V_i(S_p)$ is the maximum adjusted revenue that can be achieved from subsets of T_i when S_p is the closest offered assortment of X_i.

$$V_i(S_p) = \max \left[\max_{S_i \subseteq X_i, S_i \text{cons. w/} S_p} A(S_i \setminus S_p, S_p) + \sum_{j \in C_i} V_j(S_i), \sum_{j \in C_i} V_j(S_p) \right]. \quad (3.6)$$

This dynamic program will naturally have a higher runtime than previously. There are t stages of the dynamic program each with $O(n^w)$ possible states. In addition, solving the dynamic program requires iterating over all $O(n^w)$ possibilities for S_i and calculating $A(S_i, S_p)$, which takes at most $|\mathcal{G}|$ time. Therefore, the runtime is $O(tn^{2w}|\mathcal{G}|)$ and is still polynomial in the input size for constant tree width w.

3.4 Joint Assortment and Pricing Optimization

In the joint assortment and pricing problem, the retailer must simultaneously decide which products to offer and the prices to charge for each of these offered products with the goal of maximizing the expected revenue from each arriving customer. Recall that in this setting an arriving customer of type $g \in \mathcal{G}$ is also associated with a budget b_g and will purchase the highest ranking product that is priced within her respective budget, if any. Let $\{b_1, b_2, \ldots, b_d\}$ be the set of budgets for all customers in $\mathcal{G}$. We assume the budgets are indexed such that $b_i \leq b_{i+1}$ for all $i = 1, \ldots, d-1$.

If a customer purchases product i priced at $p \in \mathcal{R}_+$, then the retailer makes a revenue of $p - k_i$, where k_i is the fixed unit cost of acquiring one unit of product i. As shown in Lemma 2.0.1, we can restrict the possible prices of products to the set $\{b_1, \ldots, b_d, b_{d+1}\}$, where $b_{d+1} = \infty$. For example, these budgets might fall on

dollar increments. Note that the number of budgets cannot exceed the number of customer classes so $d = O(m)$. Thus, the joint assortment and pricing problem is given by

$$\mathrm{OPT}_p = \max_{p \in \{b_1, \ldots, b_{d+1}\}^n} \sum_{i \in N} (p_i - k_i) \mathrm{Pr}_i(p). \tag{3.7}$$

For the remainder of this paper, we say that product i is priced at level j to mean that $p_i = b_j$. Further, we use $r_{i,j} = b_j - k_i$ to represent the revenue when product i is priced at price level j.

It is easy to see that the joint assortment and pricing problem generalizes the standard assortment optimization problem from Section 3.2, and thus presents new difficulties. As a result, we consider the problem for two special cases. We first consider the case when the set of feasible pricing policies must be quality consistent within the tree structure, i.e., the price of any product offered is at most the price of any of its predecessors. In this case, we show that the optimal prices can be derived from the dynamic programming idea that we developed for the pure assortment problem. Second, we consider the case when all preference lists are derived from a tree that is a single path on the products. We rename this model the interval model, since the preference lists derived from the aforementioned tree structure can be viewed as intervals of consecutive integers when the products are indexed appropriately. For the interval model, we develop a novel dynamic programming approach that is completely distinct from the approach given in Section 3.2.

3.4.1 Tree Consistent Pricing

We say that prices are *tree consistent* if the price of any product offered is at most the price of any of its predecessors. We first show that this structure arises natu-

rally when customers make purchasing decisions according to the outtree model of Honhon et al. [31], in which every customer's highest ranked product is the root of the tree. Specifically, we show below that there exist optimal prices in the outtree model that are tree consistent. This matches the intuition that products towards the root are the most feature rich and should have higher prices.

Lemma 3.4.1. *Under the outtree model in which every customer's most preferred product is the root, there exists an optimal set of prices p^* to the joint assortment and pricing problem that are tree consistent.*

Proof. Suppose that the optimal solution prices product i at price level $l \leq d$ and product j at price level $k \leq d$ such that $k < l$ and j is a predecessor of i. Any customer $g \in \mathcal{G}$ such that $i \in \sigma_g$ with budget $b_g \geq b_l \geq b_k$ also must have $j \in \sigma_g$. Therefore, product i will never be purchased since if there exists $g \in \mathcal{G}$ such that $i \in \sigma_g$, then we must have $j \in \sigma_g$ and $\sigma_g(j) < \sigma_g(i)$ and thus product j will be purchased by customer class g. This means that removing product i from the assortment has no effect. We can continue removing products in this fashion until the prices are tree consistent. □

We now return to thinking about the joint assortment and pricing problem under the *general* nonparametric tree model without restricting ourselves to the outtree. For this problem, the tree consistent restriction is necessary for us to develop a tractable approach. With this restriction on the prices, we can borrow the dynamic programming ideas we use in Section 3.2 after they are translated to a pricing setting. Since our approach mirrors the dynamic programming approach from the pure assortment section, we leave out many of the technical details.

First, we develop a bit of notation for how we express our pricing decisions.

Given a matching $\mathcal{S}$ consisting of pairs (i,k) where $i \in N$ and $k \in \{1,2,\ldots,d+1\}$, we define prices $p(\mathcal{S}) = (p_1, p_2, \ldots, p_n)$ such that for $j = 1, \ldots n$

$$p_j = \begin{cases} b_k & \text{if } (j,k) \in \mathcal{S} \\ b_{d+1} & \text{otherwise.} \end{cases}$$

In other words, $p(\mathcal{S})$ sets prices according to the pairings in $\mathcal{S}$ and sets the price of all unpaired products to b_{d+1}. With this notation in place, we update our notion of blocking to account for the pricing decisions and budgets of customers. For a product i priced at level k and a product j priced at level l, we define the amount product i priced at level k blocks product j priced at level l as

$$B_{(i,k),(j,l)} = \sum_{g:\pi_g(p(\{(i,k),(j,l)\})=i,\pi_g(p(\{(j,l)\})=j} \lambda_g.$$

This captures exactly the customer classes that would have purchased product j at price level l had we not offered product i at price level k.

Consider a product i and its closest offered predecessor p priced at level l. Any customer of type g with i in her preference list that is traveling away from the root will be blocked from purchasing product i if and only if there is an offered predecessor k within her budget that she purchases. Since prices must be tree consistent and k is either p or a predecessor of p, this implies that p is also within her budget. Similarly, any customer of type g with i in her preference list that is traveling towards the root will be blocked from purchasing i if and only if there is an offered successor within her budget. However, since prices are tree consistent, any closest offered predecessor will have a price equal to that of i or lower. So to know whether or not product i is purchased we only need to know the price of i, the closest offered successors of i, the closest offered predecessor p of i, and the price level of p. The latter is the only new addition to our dynamic programming approach.

Again, our dynamic program is based on maximizing adjusted revenues in complete subtrees of T, but now we will be able to adjust the prices and therefore the revenues of products. Each stage is a product i under consideration for pricing and the two dimensional state space is a product p, possibly equal to 0, that is the closest offered predecessor of i in T and the price level l of p. Our value function $V_i(p,l)$ is the maximum adjusted revenue that can be achieved from pricing T_i when p is the closest offered predecessor of i and it is priced at level l.

$$V_i(p,l) =$$
$$\max\left\{\max_{k=1,2\ldots,l}\left[r_{i,k}\Pr{}_i(p(\{(i,k)\})) - r_{i,k}B_{(p,l),(i,k)} - r_{p,l}B_{(i,k),(p,l)} + \sum_{c\in C_i} V_c(i,k)\right],\right.$$
$$\left.\sum_{c\in C_i} V_c(p,l)\right\}.$$

The inner maximization represents setting product i at price level k, which must be at or below price level l for the prices to be tree consistent. In this case, the first two terms in this inner maximization account for the revenue generated by product i given that the closest offered predecessor to i is p offered at price level l. The third term accounts for the revenue that product i blocks from product l. Lastly, the final term is the adjusted revenues of the children's subtrees given that we priced product i at price level k. Similarly, the second part of the outer maximization accounts for when we do not offer product i.

3.4.2 Pricing with the Interval Model

In the interval model, we assume that products are indexed by decreasing quality, so that product 1 has the highest quality and product n the lowest, and that customers come in considering a *quality interval*. This quality interval represents

the range of qualities of a particular product that a customer is potentially willing to purchase. For example, we might have a customer that just considers the top quality product. In contrast, we might have another customer that will consider only mid-quality to low-quality products. This may be because the products of higher quality can sometimes come with a trade-off such as ease of use or weight. For example, an average user may not be able to easily use a high-end camera and a backpacker may not want the added bulk of high-end luggage or the possibility of it being stolen. Such a scenario could be derived from a conjunctive screening rule with lower thresholds for quality or ease. To model such a scenario in our setting, each customer class g is characterized by an arrival probability λ_g, a budget b_g, and preference list σ_g of the form $[i, i+1, \ldots, j]$ where $1 \leq i \leq j \leq n$. It is through these preference lists that we capture the quality interval of each customer class. Our algorithm extends if we also allow preference lists in opposite order, that is of the form $[i, i-1, i-2, \ldots]$. However, for ease of exposition, we ignore these potential customer classes.

The dynamic program that we develop focuses on the optimal way to price subintervals of products. As we price products, we continuously partition our intervals into smaller and smaller non-intersecting subintervals, which admit independent pricing problems. At the heart of our dynamic program formulation is the recursive manner with which we are able to stitch together the optimal pricing schemes for each interval while correctly accounting for the accrued revenue as we do so.

To further build intuition, suppose that we start by pricing product l at price level 1, the lowest price level. If we decide to price every product $k < l$ at a price level above 1, then we know that all customers $g \in \mathcal{G}$ with $l \in \sigma_g$ and $b_g = b_1$ will

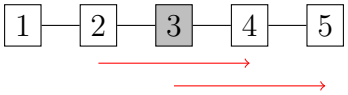


Figure 3.3: If 3 is the first product offered at the lowest budget level, then a customer with preference list $(2, 3, 4)$ will always purchase 3 if they don't purchase product 2. A customer with preference list $(3, 4, 5)$ will purchase product 3.

purchase product l irrespective of the other pricing decisions. This fully accounts for all customers purchasing product l with budget b_1. We can now consider pricing products with index smaller than l at price levels b_2 and higher. On the other hand, products indexed higher than l can still potentially be purchased at b_1 but these purchases will be made by customers whose interval starts after product l. Therefore, l acts as a breakpoint in the interval, and we can show that the problem decomposes by the products indexed higher and lower than this point. See Figure 3.3.

In particular, let $V_{i,j}(k_1, k_2)$, for all $1 \leq i, j \leq n$ and $1 \leq k_1 \leq k_2 \leq d$, be the value functions of our dynamic program. The value function represents the maximum expected revenue that can be accrued from customers whose highest ranked product is in the interval $[i, i+1, \ldots, j]$, when:

- the price level charged for product $j+1$ is k_1 (if $j = n$, we write $k_1 = \emptyset$),
- we only account for customers with a budget that is at least b_{k_2}, and
- we price each product in the interval $[i, i+1, \ldots, j]$ at a price level of k_2 or higher.

Note that $V_{1,n}(\emptyset, 1)$ gives the optimal expected revenue for the joint assortment and pricing problem. We can calculate $V_{i,j}(k_1, k_2)$ using the following dynamic

program

$$V_{i,j}(k_1,k_2) = \max\{ \sum_{g\in\mathcal{G}:\sigma_g^{-1}(1)\geq i, j+1\in\sigma_g, b_g=b_{k_2}} \lambda_g \cdot r_{j+1,k_1} + V_{i,j}(k_1,k_2+1), \quad (3.8)$$

$$\max_{l:i\leq l\leq j}[\sum_{g\in\mathcal{G}:\sigma_g(l)=1, b_g=b_{k_2}} \lambda_g \cdot r_{l,k_2} + V_{i,l-1}(k_2,k_2) + V_{l+1,j}(k_1,k_2)]\}$$

with the base cases

$$V_{i,i-1}(\cdot,\cdot) = 0 \quad \text{and} \quad V_{i,j}(\cdot,d+1) = 0.$$

The two cases in the maximum of the dynamic program above correspond to our decision of whether or not to price a product in the interval $[i, i+1, \ldots, j]$ at level k_2. The first case corresponds to not pricing any of these products at level k_2. In this case, each customer class $g \in \mathcal{G}$ such that $\sigma_g^{-1}(1) \geq i$ and $j+1 \in \sigma_g$ whose budget is b_{k_2} will purchase product $j+1$ at price b_{k_1}. For the interval $[i, i+1, \ldots, j]$, we move on to considering prices and budgets at the k_2+1 price level. In the second term, we make the decision to price one of the products in the interval $[i, i+1, \ldots, j]$ at price level k_2. The inner maximization finds the best product l to price at such a level. When we make this decision, we know that all customers $g \in \mathcal{G}$ such that $\sigma_g(l) = 1$ with budget $b_g = b_{k_2}$ will purchase product l at b_{k_2}. We then decompose the problem into disjoint intervals $[i, \ldots, l-1]$ and $[l+1, \ldots, j]$ whose optimal expected revenues can be computed separately. Since product l is priced at level k_2, we update the first entry of the state space to k_2 for the left interval. Algorithm 1 gives the recursive manner with which the value functions can be computed. Note that the ordering with which it is necessary to compute the value functions is non-trivial and is important for developing the inductive proof for Theorem 3.4.2, which proves the correctness of our dynamic program.

```
# Dictionary which stores the i, j values of the DP ;
Initialize: D = {} ;
for i = 1 to n do
    D[i] = i ;
end
while D[1] ≤ n do
    for i = 1 to n do
        j = D[i];
        for k_2 = d to 1 do
            for k_1 = k_2 to 1 do
                if j ≤ n then
                    Compute V_{i,j}(k_1, k_2);
                end
            end
        end
        D[i]+ = 1;
    end
end
```

Algorithm 1: Method for computing the value function of the DP under the interval model.

Theorem 3.4.2.

$$V_{1,n}(\emptyset, 1) = \mathrm{OPT}_p.$$

Proof. We prove the result by proving the correctness of the dynamic program given in Equation 3.8 through an inductive argument. The base cases hold trivially. We will now prove the correctness of the dynamic program for arbitrary value

function $V_{i,j}(k_1, k_2)$. The base cases give rise to the following induction hypothesis. We can assume that the value function $V_{i',j'}(k_1, k_2')$ are correctly computed for the following combinations of i', j', and k_2': $i' = i, j' = j, k_2' = k_2 + 1$ and $i \leq i', j' \leq j, k_2' = k_2$.

Suppose that we decide not to price any product in the interval $[i, i+1, \ldots, j]$ at price level k_2. Then, customers $g \in \mathcal{G}$ such that $\sigma_g^{-1}(1) \geq i$ and $j+1 \in \sigma_g$ with budget $b_g = b_{k_2}$ will purchase product $j+1$ at price b_{k_1}. From these customers we gain an expected revenue of

$$\sum_{g \in \mathcal{G}: \sigma_g^{-1}(1) \geq i, j+1 \in \sigma_g, b_g = b_{k_2}} \lambda_g \cdot r_{j+1,k_1}.$$

It remains to maximize the expected revenue from customers $g \in \mathcal{G}$ such that $i \leq \sigma_g^{-1}(1) \leq j$ and whose budget is at least b_{k_2+1}. Since we have decided not to price any of these products at price level k_2, we can now consider price levels $k_2 + 1$ and higher for such customers. By our induction hypothesis, this expected revenue is given by $V_{i,j}(k_1, k_2 + 1)$ since product $j + 1$ is still priced at level k_1. Combining both terms gives exactly the first term in the maximization of our dynamic program given in Equation 3.8:

$$\sum_{g \in \mathcal{G}: \sigma_g^{-1}(1) \geq i, j+1 \in \sigma_g, b_g = b_{k_2}} \lambda_g \cdot r_{j+1,k_1} + V_{i,j}(k_1, k_2 + 1).$$

Otherwise, let product l be priced at level k_2. Note that the inner maximization over l ensures that we choose the optimal product to price at level k_2. Then, any customer class $g \in \mathcal{G}$ such that $\sigma_g(l) = 1$ and budget $b_g = b_{k_2}$ will purchase product l. This generates expected revenue

$$\sum_{g \in \mathcal{G}: \sigma_g(l) = 1, b_g = b_{k_2}} \lambda_g \cdot r_{l,k_2}.$$

We are still left to account for the revenue accrued from customers $g \in \mathcal{G}$ such that $i \leq \sigma^{-1}(1) < l$ with budget $b_g \geq b_{k_2}$. Since product l is now priced at level k_2, we compute this revenue inductively from $V_{i,l-1}(k_2, k_2)$. On the other hand, the maximum expected revenue from customers with $l < \sigma_g^{-1}(1) \leq j$ can be found inductively from $V_{l+1,j}(k_1, k_2)$. Therefore, the overall maximum expected revenue is

$$\sum_{g \in \mathcal{G}: \sigma_g(l)=1, b_g = b_{k_2}} \lambda_g \cdot r_{l,k_2} + V_{i,l-1}(k_2, k_2) + V_{l+1,j}(k_1, k_2).$$

Taking the maximum over these two possibilities proves the claim. □

Interestingly, if we ignore fixed costs then we can find an optimal assortment in which all products are offered.

Lemma 3.4.3. *If $r_{i,j} = p_j$ for all $1 \leq i \leq n$ and $1 \leq j \leq d$, there exists an optimal assortment under the interval model that contains all products.*

Proof. Suppose that product i is not offered in the optimal assortment. Let j be the first product after i that is offered and let p_k be its price. Then consider adding product i to the assortment at price p_k. Any customer that purchases i in the modified assortment either would not have made a purchase before or would have purchased product j at the same price. Therefore, the revenue can only be improved. □

3.5 Computational Experiments

In this section, we provide computational experiments which demonstrate the efficiency of the dynamic program presented in Equation 3.3 to solve the costed

assortment problem. We benchmark ourselves against the algorithm provided for intrees in Honhon et al. [31]. This algorithm has a theoretical runtime that is exponential is the number of products, but has been been shown to work far better in practice. Since this algorithm is only valid when applied to problems in which the least preferred product of all customer types is the root node, we restrict our computational experiments to cases of this nature.

3.5.1 Experimental Setup

In our computational experiments we generate a number of intree instances to test the efficacy of our dynamic program. For each instance, we solve the costed assortment problem using two different strategies. The first strategy utilizes the dynamic program given in Equation 3.3, which we refer to as DP. The second approach uses the algorithm given in Honhon et al. [31], which we refer to as ALG3 since it is labeled Algorithm 3 in this paper. Our goal is to compare the performance of DP and ALG3 by measuring the respective CPU seconds required to solve each instance of the assortment problem.

We generate each of the intree instances in the following manner. Each of the instances that we consider consists of customer classes derived from a complete binary tree. In other words, the total number of nodes or products in each intree is $n = 2^{\mathcal{D}} - 1$ where we vary the depth of the tree as $\mathcal{D} \in \{3, 4, \ldots, 9, 10\}$. Since ALG3 is only valid when the least preferred product of each customer is the root node, we restrict the set of customer classes derived from each intree to be of this variety. For each instance, we consider all n customer types and assume that each type arrives with equal probability. So if the root node is given index n, we consider all customer types whose preference orderings take the form $\{i, \ldots, n\}\ \forall\ i \in N$.

	DP		ALG3	
$\mathcal{D}$	Avg Secs.	Max Secs.	Avg Secs.	Max Secs.
3	2.6×10^{-4}	3.2×10^{-4}	4.4×10^{-4}	1.0×10^{-3}
4	7.7×10^{-4}	8.9×10^{-4}	1.9×10^{-3}	0.017
5	2.1×10^{-3}	2.3×10^{-3}	0.011	0.089
6	5.4×10^{-3}	5.7×10^{-3}	1.00	30.78
7	0.014	0.015	9.92	262.5
8	0.033	0.035	NA	NA
9	0.081	0.084	NA	NA
10	0.19	0.20	NA	NA

Table 3.2: Comparing DP and ALG3 in terms of CPU seconds required to solve the costed assortment problem.

The revenues of each products are generated uniformly from the interval $[0, n]$. Once the revenues have been generated for a given problem instance, we then generate a fixed cost k_i for each product i uniformly over the interval $[0, r_{min}]$, where r_{min} is the smallest randomly generated revenue for the given instance. In this way, we ensure that the cost of offering a product never exceeds the revenue gained from a sale of the product. We leave out substitution costs.

3.5.2 Results

Table 3.2 summarizes our computational results. In all cases we used Python 2.7 on a Dell with an Intel Core i7-2600 Processor with 2.4 GHz and 8GB of RAM. The first column gives the number of levels in the intrees that we consider. For each value of $\mathcal{D}$, we generate 100 unique intrees using the method described in the previous section. The second column gives the average CPU seconds required for DP to solve the 100 instances, and the third column gives the maximum CPU seconds for DP over these 100 instances. Columns 4 and 5 give these same two statistics for ALG3.

The results in Table 3.2 indicate that DP significantly outperforms ALG3 in both average performance and worst case performance. Most notably, we confirm that DP does in fact scale polynomially with the number of nodes, while ALG3 appears to be on more of an exponential trajectory. Further, for DP, we observe that the maximum runtime is at most 25% larger than the average runtime over all values of $\mathcal{D}$. On the contrary, when $\mathcal{D} = 7$, the maximum runtime for ALG3 exceeded the average runtime by over 2500%. Since the maximum runtime appears to be growing exponentially with $\mathcal{D}$, it was not possible to get a sense of how ALG3 performs on the bigger instances with $\mathcal{D} > 7$. On the other hand, DP solves instances of the costed assortment problem with over 1000 products in fractions of a second.

3.6 Estimation and Analysis

In this section, we provide computational experiments that demonstrate that the nonparametric tree choice model is more effective at capturing customer behavior than the well-known MNL model in two distinct settings. In the first setting, we consider a general setup in which the retailer believes there is some structured ordering on how customers rank each product. We compare the two models on synthetic data generated from general nonparametric choice models whose underlying preference lists start as a tree in our initial test cases but become progressively more noisy and random in later test cases. In this way, we are not only able to show that we are able to accurately recapture an underlying nonparametric tree choice model, but we also show that the nonparametric tree choice model performs quite well even when the retailer's understanding of the choice process is only vaguely accurate. Our main finding is that when assortment decisions are

made based on fitted nonparametric tree models rather than fitted MNL models, the increase in profits can be as high as 20%. In the second setting, we compare the fitted nonparametric tree model versus the fitted MNL model on real sales data from two different hotels. We show that the fitted nonparametric tree model has test log-likelihood that is on average slightly higher than that of the fitted MNL model.

Building on the motivation presented in Chapter 1, we note that it is best to use the nonparametric tree choice model in settings where customers have monotonic preferences for different features in a product line, and the retailer would like to understand how customers trade off between various combinations of these features when making a purchase. For hotel bookings these features include price, discounts, bed size, square footage, the presence or absence of beautiful views, etc. The structure of the tree provides insights into how customers value these features when deciding which hotel room to purchase. In contrast, the tree model may not be appropriate in a setting in which product features are difficult to compare (e.g. horizontally differentiated features). However, in most scenarios, including this hotel example, it is unclear exactly how the products should be ordered in the tree. Herein lies the importance of our estimation procedure, which discovers this structure for us.

Our estimation procedure for the nonparametric tree model has two stages. Given a set of sales data, we first use a greedy heuristic described in the next section to construct the tree T that determines the set of feasible preference lists for our nonparametric tree model. Next, we derive the fit of the nonparametric tree model and the MNL model through maximum likelihood estimation (MLE). We use NP and ML to denote the fitted nonparametric tree model and MNL model,

respectively. In an effort to fit sparser choice models, we build the tree by only considering customer types associated with linear paths that move away from the root. In other words, we build generalized outtrees in which the most preferred product of a customer class can be any product in the tree.

3.6.1 Building and Fitting the Tree Model

We now describe how we build the tree T of products that is used to construct the preference lists of the nonparametric tree model. We assume that we have access to the past purchase history of τ customers. We represent this purchase history as the set $\text{PH} = \{(S_t, z_t) : t = 1, \ldots, \tau\}$, where S_t is the assortment of products offered to customer t and z_t is the product purchased by this customer. We set $z_t = 0$ if customer t selects the no-purchase option. We use a greedy heuristic that incrementally adds nodes to the existing tree with the goal of maximizing the number of customer classes that could have arrived in each period. Specifically, let T^i be the tree in iteration i of the greedy procedure and let $\mathcal{G}(T^i)$ be the set of customer types that can be derived from the tree T^i. For a customer that was offered assortment S and purchased product z, we say that customer class g could have arrived if $\pi_g(S) = z$. For a given tree T^i, we let

$$I(T^i) = \sum_{t=1}^{\tau} \sum_{g \in \mathcal{G}(T^i)} 1_{\pi_g(S_t) = z_t}$$

be the total number of customer classes that could have arrived over the τ customer arrivals when the set of preference lists is derived from the tree T_i. We use the function $I(T^i)$ as a proxy for how well the given tree explains the historical sales data.

Let the set $\hat{\cup}$ represent all insertions of product $j \notin T^i$ into the current tree

T_i that maintain a tree structure. In iteration $i+1$ of the greedy heuristic, we find the best way to add the next product into the tree by setting $(j^*, \cup^*) = \text{argmax}_{j \in N \setminus T^i} \text{argmax}_{\cup \in \hat{\cup}} I(T^i \cup \{j\})/|\mathcal{G}(T^i \cup \{j\})|^p$ through complete enumeration. We then set $T^{i+1} = T^i \cup^* \{j^*\}$. We continue in this manner for $n-1$ iterations, at which point we will have placed all of the products in the tree. Notice that we normalize $I(\cdot)$ by the number of customer classes that can be derived from the tree raised to a power p. By varying p, we show that we can control the depth of the tree that we discover and hence the number of arrival probabilities that need to be estimated. In our computational experiments we vary $p \in \{0, 0.5, 1\}$ and for each fitted model we report the average depth of the trees that our heuristic finds. In Section 3.6.1 we provide an example of running this greedy heuristic and show the effect of p, and in Section 3.6.3 we provide an example tree produced by our heuristic on the hotel dataset.

Now that we have built the tree from the past purchase history, we need to estimate the arrival probabilities for all possible paths. For a given tree T, the log-likelihood can be expressed as a function of the arrival probabilities λ_g for all paths in the tree T. We write the log-likelihood of the training set as $\mathcal{L}(\lambda) = \sum_{t=1}^{\tau} \log \sum_{g \in \mathcal{G}(T)} \lambda_g 1_{\pi_g(S_t)=z_t}$. Note that it is immediately obvious that the log-likelihood is concave and thus maximum likelihood estimation will be tractable. In our computational experiments, we use MATLAB's built in non-linear constrained solver $fmincon$ to get our maximum likelihood estimates. Since we estimate nonparametric choice models with at most $O(n^2)$ customer classes this approach is highly efficient.

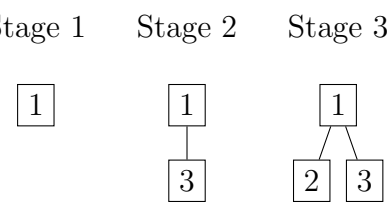


Figure 3.4: An example of building a tree greedily from purchase history $\{(\{1\}, 1), (\{1, 2, 3\}, 1), (\{2, 3\}, 2), (\{3\}, 3)\}$ with $p = 0.5$. This resulting tree is consistent with every data point in the purchase history.

Example of Building a Nonparametric Tree Choice Model from Purchase History

In this section, we give an example of greedily building a tree T from the purchase history. Let the purchase history be $PH = \{(\{1\}, 1), (\{1, 2, 3\}, 1), (\{2, 3\}, 2), (\{3\}, 3)\}$ and set $p = 0.5$. To start, we add product 1 as a singleton node since this tree describes the first two data points. Next, we can either add product 2 or product 3 to the tree. In either case, adding the product as a child of 1 would be consistent with the most amount of data points. Suppose we add product 3 as a child of 1. We are then left with adding product 2. Adding product 2 as another child of node 1 or inserting it between nodes 2 and 3 are consistent with all the data points in the purchase history. However, since adding 2 as another child of node 1 introduces fewer possible preference lists and $p = 0.5$, our algorithm will choose that method.

3.6.2 Known Ground Choice Model

In this set of computational experiments, we generate the historical sales data from a nonparametric choice model that differs significantly from our tree-based model in most of the test cases. We refer to the model that generates the sales data as

the ground truth choice model. We concede that the nonparametric tree model is perhaps not best to use in every retail scenario, as it is not some overarching choice model. However, in this section we try to show that the model is robust enough to capture purchase behavior even if the underlying choice model is not completely tree-like.

In the ground truth choice model, the set of customer types is given by $\mathcal{G} = \{1, \ldots, m\}$. The preference list and arrival probability of each customer class $g \in \mathcal{G}$ are respectively given by σ_g and λ_g. To generate the arrival probabilities $(\lambda_1, \ldots, \lambda_m)$, we set $\lambda_g = 1/m$ so that each customer class arrives with equal probability. Our approach for generating the preference lists is motivated by a setting in which a retailer sells a set of vertically differentiated products, meaning there is an overarching ordering on the qualities of the products. To start, we associate a quality interval, as described in Section 3.4, with each customer class. In order to better capture a true heterogeneous customer population, we introduce noise into the ordering of the products. Specifically, we assume that some customers drop items from their respective quality interval. In addition, we also allow for customers to have slight deviations from the overarching quality rankings. In particular, we assume that the ordering of products in the quality interval can sometimes be flipped. By including such idiosyncrasies, the preference lists of the underlying customer population differ significantly from any set that could be generated from a nonparametric tree model.

To be more specific, the following procedure was used to generate the preference list for each customer class. We assume that product 1 has the highest quality and product n has the lowest quality. For each customer class $g \in \mathcal{G}$, we first generated an initial quality interval $q_g = [i_g, \ldots, j_g]$. The most preferred product

i_g is generated uniformly from the set $\{1, \ldots, n\}$, and the least preferred product j_g is then generated uniformly from the set $\{i_g, \ldots, n\}$. We then drop each product $k \in q_g$ with probability p_d. We update q_g to be the resulting preference list. Finally, we consider $\mathcal{F}$ flip events on q_g, which are each executed with probability 0.5. If a flip event is executed, we uniformly sample a product $k \in q_g$ and flip its ordering with the product ranked immediately ahead of it. We repeat the above procedure until we have generated m unique preference lists. To ensure that we consider a diverse array of underlying ground truth choice models, we vary $(p_d, \mathcal{F}) \in \{0, 0.25, 0.5\} \times \{0, 2, 4\}$. Note that when $p_d = 0$ and $\mathcal{F} = 0$, we recover the preference lists of the interval model described in Section 3.4. We include this combination of parameters in our computational experiments to show that when the ground truth choice model is a nonparametric tree model, then we significantly outperform the MNL model.

In all of our computational experiments, we set the number of products to be $n = 10$ and the number of customer classes to be $m = 20$. Once the ground truth choice model has been generated, we then generate the historical sales data under the assumption that the purchasing behavior of each arriving customer is governed by the ground truth choice model. Recall that we assume that we have access to the past purchasing history of τ customers. We represent this purchasing history as the set $\text{PH} = \{(S_t, z_t) : t = 1, \ldots, \tau\}$, where S_t is the assortment of products offered to customer t and z_t is the product purchased by this customer. We set $z_t = 0$ if customer t selects the no-purchase option. We sample the subsets S_t such that each product is included in the assortment with probability 0.75. The class g_t that customer t belongs to is sampled from the distribution $(\lambda_1, \ldots, \lambda_m)$. Given that the ground truth choice model is a nonparametric choice model, we set $z_t = \text{argmin}_{i \in S_t} \sigma_{g_t}(i)$.

For each choice of p_d and $\mathcal{F}$, we generate 10 ground truth choice models. Then, for each of these choice models we generate 10 past purchase histories with $\tau = 2500$. Lastly, for each of these past purchase histories we generate 100 different possible revenues for the products where the revenue of each product is generated uniformly at random from the interval $[0, 100]$. This gives us $9 \times 10 \times 10 \times 100 =$ 90,000 datasets where dataset DS_k is associated with a purchase history PH_k and a set of revenues $(r_1^k, \ldots, r_n^k)$.

We test the efficacy of the fitted models by computing the optimal assortment recommended by each of the fitted models under the assumption that choice is governed by that fitted model. We then test the performance of these recommended assortments under the ground truth choice model. We also compare how well the two fitted models predict future buying behavior by computing the log-likelihoods of each fitted model on a testing set of sales data.

We first find the optimal assortments under the assumption that customer choice is governed by each of the fitted models. Suppose on a dataset DS_k generated from ground truth choice model GC, we have fitted models NP and ML. For $\text{CM} \in \{\text{NP}, \text{ML}, \text{GC}\}$, let $\text{Pr}_i^{\text{CM}}(S)$ be the probability that product i is purchased under choice model CM. We compute the optimal recommended assortment under the fitted nonparametric tree model as $S^k(\text{NP}) = \text{argmax}_{S \subseteq N} \sum_{i \in N} r_i^k \text{Pr}_i^{\text{NP}}(S)$ and the optimal assortment under the fitted MNL model as $S^k(\text{ML}) = \text{argmax}_{S \subseteq N} \sum_{i \in N} r_i^k \text{Pr}_i^{\text{ML}}(S)$. We then check the performance of these assortments by computing how well these assortments perform under the ground truth choice model, which is assumed to be reality. In particular, we compute expected revenues $R^k(\text{NP}) = \sum_{i \in N} r_i^k \text{Pr}_i^{\text{GC}}(S^k(\text{NP}))$ and $R^k(\text{ML}) = \sum_{i \in N} r_i^k \text{Pr}_i^{\text{GC}}(S^k(\text{ML}))$. For each ground truth choice model, we store

Param. ($\mathcal{F}$,	Comb. p_d)	Grnd. Choice Model # 1	2	3	4	5	6	7	8	9	10	Avg.
(0,	0)	16.86	15.39	17.19	12.86	17.56	17.28	18.03	16.27	20.89	17.69	17.00
(0,	0.25)	17.18	8.66	9.81	11.98	13.90	9.44	10.81	13.38	10.67	9.18	11.50
(0,	0.5)	13.64	10.71	12.13	12.53	13.84	11.23	10.43	13.08	15.20	14.58	12.73
(2,	0)	22.98	12.33	17.58	18.48	12.16	16.57	16.50	14.13	19.68	11.28	16.17
(2,	0.25)	12.23	14.67	11.63	13.78	15.66	11.00	10.30	8.41	7.53	15.95	12.11
(2,	0.5)	12.19	10.30	11.74	11.18	12.24	11.95	12.77	14.33	10.60	10.23	11.75
(4,	0)	16.63	11.17	15.03	12.35	14.56	15.23	10.13	9.63	17.42	11.99	13.42
(4,	0.25)	9.04	13.40	9.49	9.65	10.11	6.03	18.90	9.03	12.21	10.07	10.79
(4,	0.5)	15.21	10.80	14.09	12.83	11.60	13.31	11.42	10.70	10.09	10.65	12.07

Table 3.3: The average percentage improvement in the expected revenue of the recommended assortment of the nonparametric tree choice model over the MNL model for each ground truth choice model when $p = 0$.

Param. ($\mathcal{F}$,	Comb. p_d)	Grnd. Choice Model # 1	2	3	4	5	6	7	8	9	10	Avg.
(0,	0)	14.41	14.57	17.20	12.04	17.56	17.28	18.07	16.33	20.42	17.15	16.50
(0,	0.25)	16.97	7.05	6.59	11.98	13.90	8.84	8.35	13.78	10.67	9.19	10.73
(0,	0.5)	13.71	9.70	10.60	12.93	12.35	11.23	10.57	13.08	15.42	14.23	12.38
(2,	0)	20.59	9.10	16.14	18.47	10.68	16.26	16.48	13.55	18.34	11.28	15.09
(2,	0.25)	11.64	11.88	10.97	11.82	15.66	9.84	9.48	4.72	5.65	15.95	10.76
(2,	0.5)	12.07	9.42	7.47	11.32	11.64	11.04	12.49	12.76	10.60	11.09	10.99
(4,	0)	15.54	9.86	14.77	12.22	11.47	14.54	7.30	8.41	15.21	11.35	12.07
(4,	0.25)	7.09	13.39	9.49	9.87	8.42	3.70	17.18	6.19	10.86	10.07	9.62
(4,	0.5)	14.90	10.26	14.17	12.75	9.35	13.29	7.20	10.83	10.13	10.24	11.31

Table 3.4: The average percentage improvement in the expected revenue of the recommended assortment of the nonparametric tree choice model over the MNL model for each ground truth choice model when $p = 0.5$.

the average expected revenue of the recommended assortments over the datasets for both of the models.

Results

Tables 3.3 and 3.4 compare the predictive powers of the nonparametric tree model and the MNL model over the ninety ground truth choice models, each of which is characterized by a combinations of $\mathcal{F}$ and p_d as given in Column 1 and 2 of both tables. The numbers reported in columns 3-12 are the percentage gains,

averaged over the 100 datasets for each ground truth choice model, in the expected revenue of the recommended assortment of the fitted nonparametric tree model compared to the fitted MNL model. In other words, for the first ground truth choice model generated with $\mathcal{F} = p_d = 0$, the assortments recommended by the fitted nonparametric tree model with $p = 0$ have expected revenues that are on average 16.68% higher than the expected revenues of the assortments recommended by the fitted MNL model. Table 3.3 gives the results when $p = 0$ and Table 3.4 gives the results when $p = 0.5$. Recall that p is the power of the normalizing term in our heuristic for building the trees, and we expect that as p increases, the depth of the fitted tree decreases. This turns out to be exactly what we observe; when $p = 0$ the average depth is 10 (always builds an interval model) and when $p = 0.5$ the average depth is 8.47 with a standard deviation of 1.21 and we get quite a diverse array of trees. Generally, the trees built with $p = 0$ only perform slightly better than the trees built with $p = 0.5$, which is somewhat surprising considering that the trees built with $p = 0$ have significantly more customer types. The ability to fit trees of varying depth could be especially useful when the number of products is too large to estimate the $O(n^2)$ parameters of the interval model.

Overall, it is clear from Tables 3.3 and 3.4 that the assortments recommended by the nonparametric tree choice model are far more profitable than the assortments recommended by the MNL model. For the trees fit with $p = 0$, the average percentage gain across all parameter combinations never drops below 10%. For the trees fit with $p = 0.5$, the smallest average is 9.62%. Further, there are instances in both tables where the improvements of the nonparametric tree model exceed 20% and there is only a single instance where the average percentage improvement is below 5%.

Additionally, we compare how well the fitted nonparametric tree models and fitted MNL models perform in predicting future purchasing behavior. To do so, we compute out-of-sample log-likelihoods. For each dataset DS_k, we generate an additional $\tau = 2500$ purchase history points using the ground truth choice model. We refer to DS_k as the training set and this additional purchase history as the test set. We measure how well each fitted model on the training set predicts future behavior by computing the log-likelihood of each fitted model on the testing datasets. We use the terms out-of-sample log-likelihood and test log-likelihood synonymously.

Param. Comb.		Grnd. Choice Model #										
$(\mathcal{F},$	$p_d)$	1	2	3	4	5	6	7	8	9	10	Avg.
(0,	0)	6.08	4.71	4.23	3.68	3.60	3.02	3.87	3.97	4.63	3.52	4.13
(0,	0.25)	0.94	-0.12	1.24	1.33	2.32	1.04	0.92	1.70	0.74	2.07	1.22
(0,	0.5)	1.34	1.55	0.77	0.45	-0.82	1.34	0.81	0.20	0.27	0.72	0.66
(2,	0)	0.78	-0.40	1.41	1.22	0.52	0.93	1.30	0.05	-0.70	1.07	0.62
(2,	0.25)	-0.64	-0.18	-0.82	1.82	0.93	-0.66	-1.49	-1.53	-0.80	0.36	-0.30
(2,	0.5)	0.25	0.48	0.36	-0.39	0.34	-0.30	-0.43	0.59	-0.91	0.36	0.04
(4,	0)	-0.65	-0.52	-0.52	0.70	-1.08	0.59	0.03	-1.12	-0.34	-0.09	-0.30
(4,	0.25)	-1.00	0.63	0.60	-0.43	-0.95	-1.12	-1.64	0.07	-0.15	0.8	-0.32
(4,	0.5)	-0.74	0.08	-0.02	-0.21	0.046	0.61	0.06	-0.06	-0.39	0.015	-0.06

Table 3.5: The average percentage improvement in out-of-sample log-likelihood of the nonparametric tree over the MNL model for each ground truth choice model when $p = 0$.

Columns 1 and 2 of Tables 3.5 and 3.6 give the parameter combinations of $\mathcal{F}$ and p_d that dictate how the 10 ground truth choice models are generated. For each ground truth choice model, we have 1000 generated datasets. The numbers reported in columns 3-12 are the percentage gains, averaged over the datasets, in out-of-sample log-likelihood of the fitted nonparametric tree model over the fitted MNL model. Negative numbers indicate that the MNL model outperformed the nonparametric tree model using the metric of test log-likelihood. Overall, it is clear that the two choice models perform on virtually level footing. When the number of flips events is small, the nonparametric tree model outperforms the MNL model

Param. Comb.		Grnd. Choice Model #										
$(\mathcal{F},$	$p_d)$	1	2	3	4	5	6	7	8	9	10	Avg.
(0,	0)	4.81	4.56	4.23	4.62	3.60	3.02	3.90	3.94	4.60	3.38	4.07
(0,	0.25)	1.00	-0.50	0.90	1.33	2.31	0.92	0.80	1.77	0.74	2.08	1.14
(0,	0.5)	1.36	1.24	0.66	0.67	-0.86	1.34	2.14	0.20	0.45	0.86	0.81
(2,	0)	0.36	-0.99	1.17	1.27	0.13	1.44	1.31	0.18	-0.85	1.07	0.51
(2,	0.25)	-0.72	-0.94	-0.82	0.48	0.93	-0.64	-1.47	-1.83	-0.97	0.36	-0.56
(2,	0.5)	0.25	0.44	0.01	-0.20	0.28	-0.48	-0.06	-0.03	-0.89	0.39	-0.03
(4,	0)	-0.77	-0.71	-0.43	0.89	-1.17	0.53	-0.28	-1.20	-0.69	-0.13	-0.4
(4,	0.25)	-1.17	0.70	0.60	-0.40	-0.92	-1.33	-2.18	-0.32	-0.25	0.80	-0.45
(4,	0.5)	-0.69	0.04	-0.01	-0.04	0.24	0.61	-1.44	-0.05	-0.39	-0.01	-0.18

Table 3.6: The average percentage improvement in out-of-sample log-likelihood of the nonparametric tree over the MNL model for each ground truth choice model when $p = 0.5$.

almost uniformly across all of the generated ground truth choice models. However, when we generate ground truth choice models in which there are up to four flip events, the MNL performs slightly better. This is to be expected, since the ground truth choice model moves farther and farther away from a tree-like structure as the number of flip events and deletions increases. Finally, it worth noting that the trees built with $p = 0.5$ perform at least as well as the trees built with $p = 0$ more than one-third of the time.

3.6.3 Hotel Dataset

In this section, we compare the nonparametric tree model and the MNL model on the hotel bookings dataset provided in Bodea, Ferguson, and Garrow [8]. This dataset consists of bookings at 5 different hotels made from March 12, 2007 to April 15, 2007 made primarily by business customers through online channels or customer relationship employees. We decide to focus only on Hotels 1 and 3 since the number of purchases in the sales data at these two hotels exceeds the purchases at the other three hotels by a factor of five. Each hotel offers a variety of rooms

(suite, king, queen, etc.) at differing rates based on additional accommodations that the room might come with. For example, Hotel 1 offers a King Room both at Rate 1 and at Rate 5. The former is a discounted advanced purchase rate and the latter is a rate that includes city activities such as dining and shopping. Since the price for a specific room varies by its accompanying rate, we treat each room type and rate tuple as a different product. We discard products that have fewer than 5 purchases throughout the selling horizon. This notion of a product differs from the works of van Ryzin and Vulcano [58] and van Ryzin and Vulcano [59], who also use this data to fit nonparametric choice models. In these works, a product is simply a room type, and then, in order to capture the effect of the various rates, the authors assume that customers always buy up within the same room type. The upside of our approach is that we capture a more granular view of choice. However, in modeling a product as a room type-rate tuple we have more products and hence more parameters to estimate. To control for overfitting, we perform a rigorous 10-fold cross validation procedure, which we describe later in this section.

The dataset provides detailed information about the set of products that were offered and the product that was purchased at the time of each booking. Consistent with the other work that uses this dataset, we restrict our study to bookings for which there is at least one transaction per product and for which the observed purchase comes from the available options. Since the dataset only gives booking information, there are no data points where the no-purchase option is selected. To make up for this deficiency in the dataset, for each booking record we generate $np = \{0, 1, \ldots, 10\}$ no-purchase records. In this way, we can study the performance of the various choice models for varying levels of no-purchase tendencies. Table 3.7 summarizes the data availability at each of the hotels.

	Hotel #	
	1	3
# products	33	30
# data points	1267	1109

Table 3.7: The number of products and bookings with a purchase ($np = 0$) for each hotel.

We index each of the datasets using the the tuple $(h, np) \in \{1, 3\} \times \{0, \ldots, 10\}$, where the first entry h corresponds to the hotel that we consider and the second entry np gives the number of additional no-purchase bookings we add for each booking record. For each dataset, we perform 10-fold cross validation to compare the out-of-sample performance of the MNL model to the nonparametric tree model. To do so, we randomly partition each dataset into ten equal segments. Nine of the ten segments make up the training dataset, while the remaining segment is used for testing. For each test case (h, np), we build three trees by varying the normalizing constant $p \in \{0, 0.5, 1\}$. Recall that p controls the depth of the tree built. We fit the MNL model and the three general trees using MLE on the training dataset and then measure the accuracy of the fitted model using the log-likelihood of the testing sets. Of the three nonparametric tree choice models that we fit, we only test the tree model that has the largest training log-likelihood. We repeat this procedure ten times so that each segment is the testing set at one point. Then, for each dataset, we repeat this 10-fold cross validation ten times to ensure that our results are robust to the randomization that occurs within the cross validation. We average the test log-likelihoods over the ten trials and the ten folds.

Example Nonparametric Tree Choice Model Built for Hotel 1

In this section, we give the tree that our heuristic built for Hotel 1 when $np = 5$. Since this hotel has 32 products, we do not display the entire tree. Instead, we drop products from the tree that were purchased fewer than 10 times, since these products only obfuscate higher level buying trends that can be teased out from the tree. After this filtering, we then drop any remaining orphans from our displayed tree. The resulting tree, given in Figure 3.5, provides some interesting insights with regards to how customers substitute between the various rooms. For Hotel 1, Rate 1 is an advanced rate, Rate 2 is the rack rate, and Rate 3 is the rack rate combined with additional hotel services.

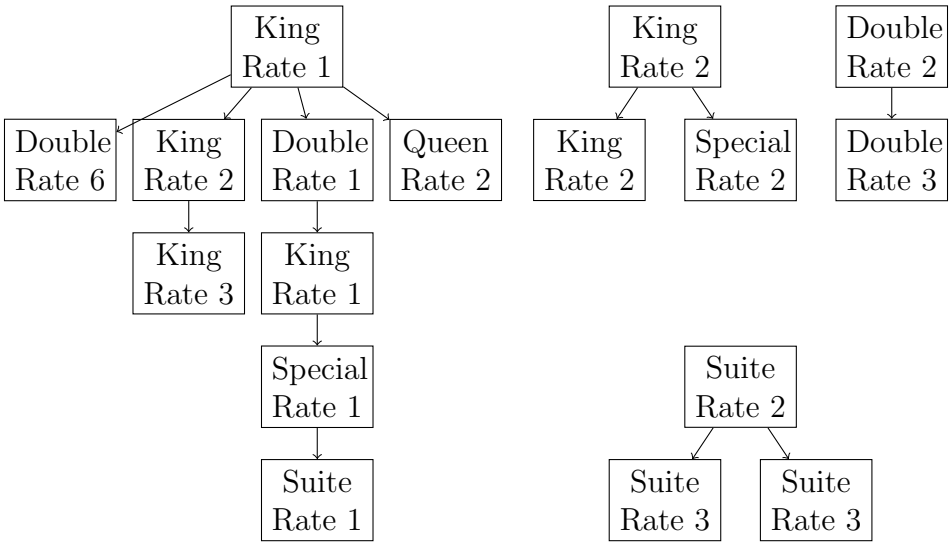


Figure 3.5: A subtree of the outtree constructed from the hotel purchase data.

The tree in Figure 3.5 reveals 4 distinct customer segments. The three subtrees on the right consist of customers interested in suites, doubles and kings respectively. The farthest left subtree describes the customers who are more flexible and are generally willing to substitute from king rooms to double or queens and then to

suites. Given that Rate 1 represents customers buying at an advanced discount rate, it is consistent with the notion that these customers might be cost-conscious that these customers consider suites last. Unfortunately, the authors of Bodea, Ferguson, and Garrow [8] never fully detail what is meant by a "Special" room and hence it is difficult to expound much on the trends regarding these rooms. Additionally, when the room type is held constant, we observe that Rate 1 and Rate 2 are generally preferred to Rate 3. This trend might indicate that the hotel is overpricing these additional hotel services, especially considering that we observe this trend in the subtree for suites.

Results

Table 3.8 gives the average depth of the trees fit for the two hotels with $p \in \{0, 0.5, 1\}$. We average the depth of the fitted trees over the eleven values of np that we test. As was intended, we see that the depth of the fitted trees decreases as we increase the normalization constant p. The percentage in parentheses given next to the average depth in this table is the fraction of test cases where the tree built with the given value of p had the largest train likelihood and thus was selected to be evaluated on the test set. Notice that the trees built with $p = 1$ are quite often the best fitting trees, which is fairly surprising since the trees fit with $p = 0$ have significantly more customer types. These results are in stark contrast to our results for the synthetic data presented in the previous section in which the trees built with $p = 0$ generally performed the best.

We turn to the underlying structure of the ground truth choice model in an effort to explain this trend. In the first set of experiments, the ground truth choice model is assumed either to be an interval model, or some slight perturbation of

	p		
Hotel #	0	0.5	1
Hotel 1	28.1 (9%)	18.2 (10%)	9.72 (81%)
Hotel 3	20.4 (9%)	14.7 (14%)	6.7 (77%)

Table 3.8: Average depth of the fitted trees for each value of np for the two hotel datasets.

the interval model. The hope in fitting a wider tree with fewer customer types but greater heterogeneity in the preference lists, is that one would be able to better capture the heterogeneity that results from the various perturbations to the interval model. On the other hand, in choosing to fit a tree with greater depth, one will likely better capture the notion that there is some underlying universal ranking of the products that is guiding purchasing behavior. Since the trees with greater depth perform better in the first set of experiments, it is clear that this latter point is more relevant; the tree structure closest to the ground truth choice model is generally a line graph. In this second set of experiments, we do not have access to the ground choice model governing purchasing decisions. Nonetheless, the fact that wider trees perform better in this setting indicates that a model that is able to capture a diverse array of preferences is more fitting than one that assumes a universal ordering of the products.

The average percentage improvements in log-likelihood of the fitted nonparametric tree model over the fitted MNL choice model are given in Table 3.9. For lower values of np, the fitted MNL model provides slight improvements over the fitted nonparametric tree choice model. However, as np increases the nonparametric tree choice model appears to dominate. Even though we improve over the MNL model by only a few fractions of a percent, these results are nonetheless surprising in light of the performance of the most general nonparametric model on these same datasets presented in past studies. Specifically, the work of van Ryzin and

	# of addition no-purchase events										
Hotel #	0	1	2	3	4	5	6	7	8	9	10
Hotel 1	-1.93	-1.08	-0.32	-0.03	-0.04	**0.27**	**0.36**	**0.33**	**0.41**	**0.49**	**0.49**
Hotel 3	**0.49**	-1.21	**0.22**	**0.24**	-0.56	-0.03	-0.02	**0.08**	**0.20**	**0.14**	**0.25**

Table 3.9: Percentage improvement in log-likelihood of the nonparametric tree choice model over the MNL model on the hotel datasets.

Vulcano [59] shows that the MNL model outperforms the general nonparametric model for all 5 hotels on the metric of AIC_c, a likelihood-based metric which is normalized by the number of parameters fitted. These results show that there can be benefits to imposing specific sparse structures on the set of preference lists in settings in which there is clearly some notion of vertical differentiation among the products. The tree model seems to be a valid candidate for imposing this structure.

CHAPTER 4

LIMITED SUBSTITUTION CHOICE MODEL

In this chapter, we study a setting in which customers are only willing to consider a limited number of products. This setting occurs naturally when there is either a high purchasing bias (e.g. customers looking for a specific feature) or a low cost of leaving the system and not making a purchase (e.g. the option to go to another retailer easily). We introduce the k-product nonparametric choice model in which all preference lists have length at most k. Interestingly, we show that this model is robust to the choice of k in that the retailer does not have to know exactly how many products customers actually consider to accurately capture purchase behavior. We show that while the assortment optimization problem under this model is NP-hard, even for $k = 2$, we are able to develop a novel approximation scheme for this optimization problem. Through a series of computational experiments, we show that the proposed algorithm is easy to implement, efficient to run, and performs significantly better than its theoretical guarantee. Further, we can extend the algorithm to the cardinality constrained assortment optimization problem under the 2-product nonparametric choice model.

4.1 Model

In the k-product nonparametric choice model, we assume that $|\sigma_g| \leq k$ for all customer classes $g \in \mathcal{G}$. Without loss of generality, we can assume $|\sigma_g| = k$ by adding in dummy products with zero revenue for customers with shorter preference lists. For each customer class $g \in \mathcal{G}$, let g_i be the ith most preferred product for $i \in \{1, 2, \ldots, k\}$. Under the k-product nonparametric choice model, the probability

product j is purchased under assortment $x \in \{0,1\}^n$ is

$$\Pr{}_j(x) = \sum_{i=1}^{k} \sum_{g \in \mathcal{G}: g_i = j} \lambda_g (1 - x_{g_1})(1 - x_{g_2}) \dots (1 - x_{g_{i-1}}) x_j.$$

When $k = 1$, every customer only has a first choice product, and since there is no substitution behavior, it is trivially optimal for the retailer to offer every product. However, as k increases both the estimation and assortment optimization problems become more difficult. The estimation problem becomes more difficult because the number of possible preference lists grows exponentially in k, and thus when k is large, it becomes computationally burdensome to derive the fitted model. Similarly, the assortment optimization problem becomes harder to solve as k increases since there is a more delicate trade-off between gaining new customers and having some customers switch products. This is further demonstrated by the degree of the polynomial in $\Pr_j(x)$.

Fortunately, in Section 4.6, we show that the k-product nonparametric model for $k \in \{2, 3, 4\}$ performs quite well even when customers substitute up to two or three times as much in reality. We also observe that increasing k beyond 4, and therefore increasing the number of possible preference lists, significantly increases the runtime of the estimation procedure while only marginally improving prediction accuracy. As a preview, we illustrate this trade-off for a single test case below. Figure 4.1 shows the relative mean average error (MAE) and runtime for the k-product nonparametric choice model (denoted NP_k) for various values of k and τ, which is the number of data points used to fit the model. We fully detail the computational results in Section 4.6.

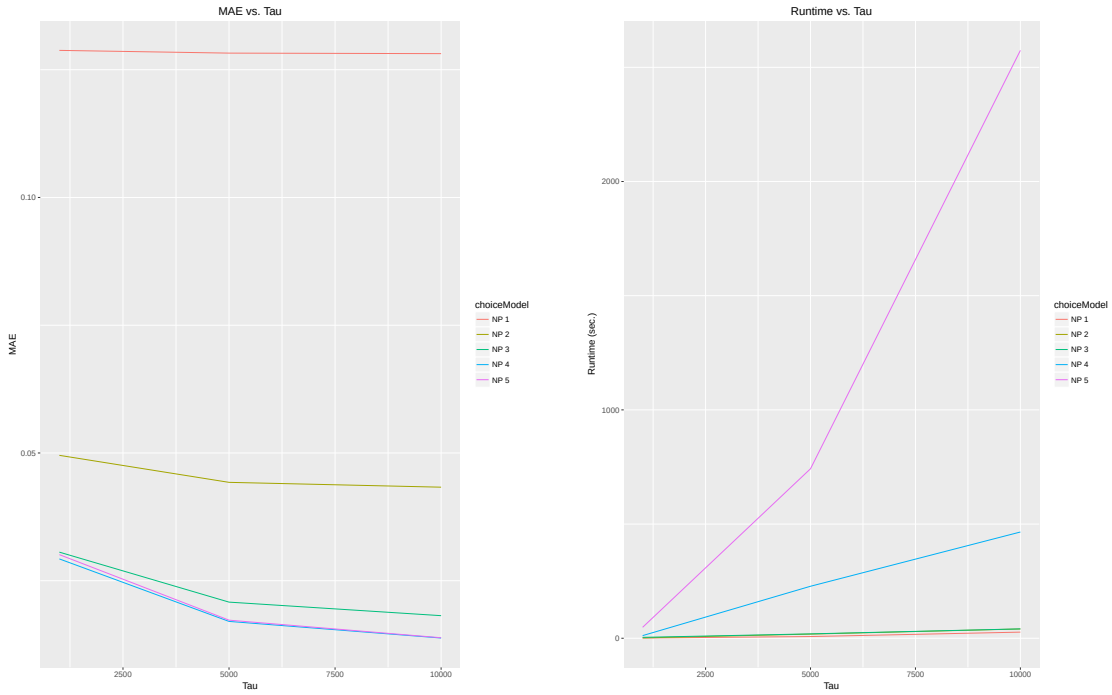


Figure 4.1: Results for each nonparametric fit. We observe that NP_5 only marginally improves upon NP_3 and NP_4 in MAE while having a significantly larger runtime.

4.2 Assortment Optimization under the 2-Product Choice Model

In this section, we first consider the assortment optimization problem under the 2-product nonparametric choice model. We then develop an alternative algorithm for the general k-product nonparametric choice model in Section 4.3. However before we develop our approximation algorithms, we first show that this problem is strongly NP-hard even for $k = 2$. Note that the hardness result for $k = 2$ implies the problem is also strongly NP-hard for $k > 2$ since the same reduction will hold by adding in dummy products.

Theorem 4.2.1. *Unless P=NP, there does not exist a fully polynomial-time approximation scheme (FPTAS) for the assortment optimization problem under the*

2-product nonparametric choice model even when all preference lists come from a single universal ordering of the products.

Proof. Consider an instance of vertex cover on a cubic graph $G = (V, E)$ with $V = \{v_1, v_2, \ldots, v_n\}$ and constant k. We construct an instance of the assortment optimization problem by creating products $\{1, 2, \ldots, n+1\}$ where $r_i = 1$ if $i \leq n$ and $r_i = 2$ if $i = n+1$. For every edge $(v_i, v_j) \in E$, we create a customer class g that first considers product $g_1 = \min(i, j)$ and then considers product $g_2 = \max(i, j)$. We refer to these customers as "edge" customers. Similarly, for each vertex $v_i \in V$, we create a customer class g that first considers product $g_1 = i$ and then considers product $g_2 = n + 1$. We refer to these customers as "vertex" customers. Edge customers will arrive with unnormalized probability 1 and vertex customers will arrive with unnormalized probability $\frac{1}{3}$. It is without loss of generality that we consider unnormalized arrival probabilities.

We will first show that there exists a vertex cover of size $\leq k$ in G if and only if the optimal assortment of products in the corresponding assortment optimization problem generates an expected revenue of at least $|E| + \frac{1}{3}(k + 2(n - k))$.

Suppose there exists a vertex cover $U \subseteq V$ such that $|U| \leq k$. Then, let $S = \{i : v_i \in U\} \cup \{n + 1\}$. For every edge customer (v_i, v_j) either v_i or v_j is in U so either i or j is in S, and thus we accumulate an expected revenue of 1 from each edge customer. Next, every vertex customer corresponding to $v_i \in U$ generates an expected revenue of $\frac{1}{3}$ since these customers will purchase product i. On the other hand, every vertex customer corresponding to $v_i \notin U$ generates an expected revenue $\frac{2}{3}$ since these customers purchase product $n + 1$. Thus, $\text{Rev}(S) = |E| + \frac{1}{3}(|U| + 2(n - |U|)) \geq |E| + \frac{1}{3}(k + 2(n - k))$.

Let S^* be the optimal solution to the assortment problem and assume that $\text{Rev}(S^*) \geq |E| + \frac{1}{3}(k + 2(n-k))$. Without loss of generality, S^* contains product $n+1$ since adding this product can only increase the revenue. Suppose that there exists an edge customer (v_i, v_j) such that neither i nor j are in S^*. Then, by adding i to S^* we gain an expected revenue of at least 1 from this customer. However, now the vertex customer associated with vertex v_i also purchases product i yielding expected revenue $\frac{1}{3}$ instead of product $n+1$ with expected revenue $\frac{2}{3}$. Overall the net gain in revenue is at least $\frac{1}{3}$ which contradicts the optimality of S^*. This shows that all edge customer classes must make a purchase when S^* is offered and thus S^* will be a vertex cover. Furthermore, we know that

$$\begin{aligned}\text{Rev}(S^*) &= |E| + \frac{1}{3}(|S^*| + 2(n - |S^*|)) \\ &\geq |E| + \frac{1}{3}(k + 2(n-k)),\end{aligned}$$

where the inequality follows from our assumption on the value of $\text{Rev}(S^*)$. Thus, we must have that $|S^*| \leq k$, and we have found a vertex cover of size $\leq k$.

In particular, this shows that if k is the size of a minimum vertex cover then the optimal revenue in the assortment problem that we create above is $\text{OPT} = |E| + \frac{1}{3}(k + 2(n-k))$. Let S_2 be an assortment that achieves the second largest revenue $\text{Rev}(S_2) = \text{OPT}_2 < \text{OPT}$ for the assortment problem. Suppose S_2 is a vertex cover in the original vertex cover problem. Then, $\text{OPT}_2 = |E| + \frac{1}{3}(|S_2| + 2(n - |S_2|)) < |E| + \frac{1}{3}(k + 2(n-k))$ since $\text{Rev}(S_2) < \text{OPT}$. This implies that $\text{OPT}_2 \leq |E| + \frac{1}{3}(k + 1 + 2(n - k - 1))$. Now suppose that S_2 is not a vertex cover. By the argument made above, if more than two edge customers do not make a purchase, then we can improve the revenue while maintaining that S_2 is not a valid vertex cover by adding one more product to the assortment. Therefore, S_2 must have exactly one edge customer not making a purchase, which implies

that $|S_2| \geq k-1$ since the minimum vertex cover has size k. Thus, $\text{OPT}_2 \leq |E| - 1 + \frac{1}{3}(k-1+2(n-k+1)) < |E| + \frac{1}{3}(k+1+2(n-k-1))$. Combining these two cases we get that $\text{OPT}_2 \leq |E| + \frac{1}{3}(k+1+2(n-k-1))$. Thus,

$$\begin{aligned}
\frac{\text{OPT}_2}{\text{OPT}} &\leq \frac{|E| + \frac{1}{3}(k+1+2(n-k-1))}{|E| + \frac{1}{3}(k+2(n-k))} \\
&= 1 - \frac{1/3}{3n/2 + \frac{1}{3}k + \frac{2}{3}(n-k)} \\
&\leq 1 - \frac{1/3}{3n/2 + \frac{1}{3}n + \frac{2}{3}n} \\
&= 1 - \frac{1}{7.5n}
\end{aligned}$$

where the equality comes from the fact that G is a cubic graph and $|E| = 3n/2$. Thus, if one can produce an assortment that is within a factor of $1 - \frac{1}{7.5n}$ of OPT, then this assortment must actually be optimal.

Suppose there existed an FPTAS for the assortment problem under the 2-product nonparametric choice model. For any input to the vertex cover problem on a cubic graph, we can transform the problem into an assortment optimization problem as above and set $\varepsilon < \frac{1}{7.5n}$. Note that $1/\varepsilon = \text{O}(n)$. Then, the FPTAS would be guaranteed to return the optimal solution, which in turn will reveal the optimal solution to the vertex cover problem in polynomial time. This is a contradiction since vertex cover on cubic graphs is NP-hard. □

Motivated by this hardness result, we turn our focus to finding an efficient approximation algorithm for this problem that performs well in practice. Interestingly, there exist several simple approximation algorithms to solve this problem. We present three methods including a reduction to a well-known discrete optimization problem, a linear-time algorithm based on the structure of the revenue function, and a linear programming based algorithm. The latter is the only method we can extend for longer preference lists and is presented in Section 4.3.

4.2.1 Reduction to Maximum Directed Cut

Lemma 4.2.2. *The assortment optimization problem under the 2-product non-parametric choice model can be reduced to an instance of maximum directed cut.*

Proof. In the maximum directed cut problem, we are given a directed graph $G = (V, E)$ along with edge weights $w_e \geq 0$ for all $e \in E$. The goal is to find a cut $S \subseteq V$ that maximizes the weight of edges directed from S to $V - S$.

Consider an instance of the assortment optimization problem in which each customer is willing to consider exactly two products. We construct a corresponding directed graph G. First, for each product $i \in N$, we construct a corresponding vertex v_i. We also add a special vertex v_0. Next, for each customer class $g \in \mathcal{G}$ with first choice product g_1 and second choice product g_2, we construct a directed edge (v_{g_1}, v_0) of weight $\lambda_g r_{g_1}$ and a directed edge (v_{g_2}, v_{g_1}) of weight $\lambda_g r_{g_2}$.

Let $S \subseteq V$ be an optimal directed cut. Without loss of generality, $v_0 \notin S$ since v_0 has no outward edges and it can only improve the solution to remove v_0 from S. For any customer class $g \in \mathcal{G}$, the edge (v_{g_1}, v_0) of weight $\lambda_g r_{g_1}$ is in the cut if and only if $v_{g_1} \in S$. Similarly, the edge (v_{g_2}, v_{g_1}) of weight $\lambda_g r_{g_2}$ is in the cut if and only $v_{g_1} \notin S$ and $v_{g_2} \in S$. Thus, the weight of the cut S is equal to the revenue of the corresponding assortment. Similarly, we can show that the revenue for any assortment $S \subseteq N$ is equal to the weight of the corresponding cut with all products in S in the cut.

Thus, finding the maximum directed cut is equivalent to finding an optimal assortment. □

Theorem 4.2.3. *There exists a 1.14-approximation algorithm for the assortment*

optimization problem under the 2-product nonparametric choice model.

Proof. Lewin, Livnat, and Zwick [40] prove a 1.14-approximation algorithm for the maximum directed cut problem which gives the result. This algorithm is an extension of the semidefinite programming algorithm proposed by Goemans and Williamson [27]. □

The algorithm in Theorem 4.2.3 rounds the solution to a semidefinite program, and it can be derandomized using the techniques proposed in Mahajan and Ramesh [43].

4.2.2 Submodularity

Solving a semidefinite program is computationally intensive. Fortunately, this problem has nice some structure that we can exploit to get a linear time approximation algorithm. In particular, we can show that the revenue function is *submodular*. Recall, a function $f : 2^N \mapsto \mathbb{R}$ is submodular if for all $S, T \subseteq N$ such that $T \subset S$ and $x \in N \setminus S$,

$$f(S \cup \{x\}) - f(S) \leq f(T \cup \{x\}) - f(T).$$

These functions capture the notion of diminishing returns. In our case, this implies that as the offered assortment grows in size the benefit of adding a product $i \in N$ decreases.

Lemma 4.2.4. *For the 2-product nonparametric choice model,* $\mathrm{Rev}(\cdot)$ *is a non-monotone submodular function.*

Proof. First, we show that $\text{Rev}(\cdot)$ is submodular. Let $S, T \subseteq N$ such that $T \subset S$ and let $i \in N \setminus S$. For any $W \subseteq N$, we define the adjusted revenue of i to be

$$A(i, W) := \text{Rev}(W \cup \{i\}) - \text{Rev}(W).$$

We show that $A(i, T) \geq A(i, S)$, which is equivalent to showing the submodularity of $\text{Rev}(\cdot)$. To do so, we consider the change in the revenue when product i is added to the assortments S and T. For each $g \in \mathcal{G}$ such that $i \in \sigma_g$, we analyze this revenue change based on the inclusion or exclusion of $j = \sigma_g \setminus \{i\}$ in S and T:

- If $j \notin S$, then adding i to S or T generates an additional revenue of $\lambda_g r_i$.
- If $j \in S$ and $j \in T$, then adding i to S or T changes the revenue by $\lambda_g(r_i - r_j)$ if $g_1 = i$ and does not change the revenue otherwise.
- If $j \in S$ and $j \notin T$, then adding i to S changes the revenue by $\lambda_g(r_i - r_j)$ if $g_1 = i$ and does not change the revenue otherwise. On the other hand, adding i to T generates additional revenue $\lambda_g r_i$.

Note that we cant have $j \notin S$ and $j \in T$ since $T \subset S$. Therefore, for each $g \in \mathcal{G}$, we have shown that the net gain in revenue for assortment T when product i is added is at least the net gain in revenue for assortment S when product i is added. Overall, this implies that $A(i, T) \geq A(i, S)$, as desired.

A function $f : 2^N \mapsto \mathbb{R}$ is monotone if for every $T \subseteq S$ we have that $f(T) \leq f(S)$. We show that $\text{Rev}(S)$ is non-monotone with a very simple two product example. Let $N = \{1, 2\}$ with $r_1 = 1$ and $r_2 = 2$. Consider a single customer class with $g_1 = 1$ and $g_2 = 2$, who arrives with certainty. In this case, $\text{Rev}(\{2\}) = 2$ and $\text{Rev}(\{1, 2\}) = 1$ and hence $\text{Rev}(S)$ is non-monotone.

This proof does not extend for longer lists. For example, consider a simple three product example. Let $N = \{1, 2, 3\}$ with $r_1 = 1$, $r_2 = 2$, and $r_3 = 3$. Further, consider a single customer class with $g_1 = 1$, $g_2 = 2$, and $g_3 = 3$. Let $T = \{3\}$ and $S = \{1, 3\}$. Then adding $j = 2$ to T decreases the expected revenue from $r_3 = 3$ to $r_2 = 2$. However, adding j to S does not affect the expected revenue. This breaks the submodularity property. □

Maximizing a non-monotone submodular function has been studied thoroughly (see [20], [10], [33], and [9]). Buchbinder et al. [10] present a randomized 2-approximation algorithm for this problem, matching the known hardness of Feige, Mirrokni, and Vondrák [20]. This result was then improved by Buchbinder and Feldman [9] who present a deterministic 2-approximation algorithm. We will present a modified version of the *randomized* algorithm of Buchbiner et al. [10] that will run in time linear in the number of customer classes and is a *deterministic* 2-approximation algorithm. In addition, we are able to slightly strengthen their result by showing that we are within a factor of 2 of the optimal value to a linear programming relaxation of the assortment problem. This revised analysis is similar to the one taken by Poloczek et al. [50] in presenting a deterministic 4/3-approximation algorithm for MAX SAT.

First, we develop some intuition. The linear time algorithm is a two-stage algorithm. In the first stage, we process products $1, 2, \ldots, n$ in order and assign each a probability $p_i \in [0, 1]$ of being in the assortment. These probabilities are set such that if each product i is offered independently with probability p_i, then the expected revenue will be at least $1/2$ the optimal value for a linear programming relaxation for the problem. Then, in the next stage of the algorithm, we derandomize the assortment using the classic method of conditional expectations

to determine whether or not to offer each product.

Given our submodular revenue function Rev : $2^N \mapsto \mathbb{R}^+$, the multilinear extension of Rev(S) is given by $F : [0,1]^N \mapsto \mathbb{R}^+$ where the value of F at $x \in [0,1]^N$ is

$$\begin{aligned} F(x) &:= \sum_{S \subseteq N} \prod_{i \in S} x_i \prod_{j \notin S} (1 - x_j) \text{Rev}(S) \\ &= \sum_{g \in \mathcal{G}} \lambda_g r_{g_1} x_{g_1} + \lambda_g r_{g_2} (1 - x_{g_1}) x_{g_2}. \end{aligned}$$

The multilinear extension computes the expected revenue when each product i is offered independently with probability x_i. We will use this multilinear extension to inform our decisions about the products.

In the first stage, the algorithm processes the products in order $1, 2, \ldots, n$ and assigns each product an offer probability. Suppose that at the beginning of iteration i, we have set probabilities $p_1, \ldots, p_{i-1}$ for products $1, 2, \ldots, i-1$. Then, let

$$x_j^{i-1} := \begin{cases} p_j & j \leq i-1 \\ 0 & j \geq i \end{cases} \quad \text{and} \quad \bar{x}_j^{i-1} := \begin{cases} p_j & j \leq i-1 \\ 1 & j \geq i \end{cases}.$$

These two vectors represent the two extremes for the remaining probabilities: either offering or not offering all of the products with index i or higher. We will keep track of the average expected revenue between these two vectors by defining

$$B_{i-1} := \frac{1}{2} \left[F(x^{i-1}) + F(\bar{x}^{i-1}) \right].$$

The algorithm essentially chooses to offer product i with probability p_i to balance between these two extremes for the remaining probabilities. Let e_j be the unit vector with a one in the jth entry. To set this probability we define

$$a_i := F(x^{i-1} + e_i) - F(x^{i-1}) \quad \text{and} \quad b_i := F(\bar{x}^{i-1} - e_i) - F(\bar{x}^{i-1}).$$

Note that a_i is the change in the expected revenue if we offer product i under the extreme that all future products will remain not offered and b_i is the change in the expected revenue if we do not offer product i under the extreme that all future products are offered. If $a_i \leq 0$, the algorithm sets $p_i = 0$ and does not offer product i. Otherwise, if $b_i \leq 0$ and $a_i > 0$, then the algorithm sets $p_i = 1$ and offers product i. Lastly, if $a_i, b_i > 0$, then the algorithm sets $p_i = a_i/(a_i + b_i)$. Note that the differences in the definitions of a_i and b_i will only contain terms for customer classes that contain i so we only need to look at the *incident* customer classes to compute these two values. In particular, a_i and b_i can be written as

$$a_i = \sum_{g:g_1=i} \lambda_g r_i + \sum_{g:g_2=i} \lambda_g r_i (1 - x_{g_1}^{i-1}) - \sum_{g:g_1=i} \lambda_g r_{g_2} x_{g_2}^{i-1}$$

and

$$b_i = - \sum_{g:g_1=i} \lambda_g r_i - \sum_{g:g_2=i} \lambda_g r_i (1 - \bar{x}_{g_1}^{i-1}) + \sum_{g:g_1=i} \lambda_g r_{g_2} \bar{x}_{g_2}^{i-1}.$$

Therefore, the first stage of the algorithm runs in linear time with respect to the number of customer classes.

The second stage of the algorithm derandomizes the probabilities p_i without decreasing the expected revenue. In this stage, we will make our final decisions about whether to offer each product. Suppose that at the beginning of the ith iteration, we have decided to offer products $S_{i-1} \subseteq \{1, 2, \ldots, i-1\}$. Then, let

$$q_j^{i-1} := \begin{cases} 0 & j \leq i-1 \text{ and } j \notin S_{i-1} \\ 1 & j \leq i-1 \text{ and } j \in S_{i-1} \\ p_j & j \geq i \end{cases}.$$

This represents the current probabilities of each product being offered. Note that the expected revenue given these current probabilities can be written as

$$F(q^{i-1}) := p_i F(q^{i-1} + (1 - p_i)e_i) + (1 - p_i) F(q^{i-1} - p_i e_i).$$

If $F(q^{i-1} + (1 - p_i)e_i) \geq F(q^{i-1} - p_i e_i)$, we offer product i. Otherwise, we do not.

Again, we can simplify the calculations to determine if $F(q^{i-1} + (1 - p_i)e_i) \geq F(q^{i-1} - p_i e_i)$. Any customer class that does not contain i will contribute the same amount to each side. On the other hand, if a customer class has $g_1 = i$ it contributes $\lambda_g r_i$ to the left hand side and $\lambda_g r_{g_2} q_{g_2}^{i-1}$ to the right hand side. If $g_2 = i$, then it contributes $\lambda_g r_{g_1} q_{g_1}^{i-1}$ to both and $\lambda_g r_i (1 - q_{g_1}^{i-1})$ to the left hand side. In other words, if

$$\sum_{g:g_1=i} \lambda_g r_i + \sum_{g:g_2=i} \lambda_g r_i (1 - q_{g_2}^{i-1}) \geq \sum_{g:g_1=i} \lambda_g r_{g_2} q_{g_2}^{i-1},$$

we offer product i and otherwise we do not.

Overall, this shows the algorithm runs in linear time with respect to the number of customer classes. We summarize this two stage linear-time algorithm in Algorithm 2.

The following lemma about the probability values will be useful in the later analysis.

Lemma 4.2.5. *For all $i = 1, 2, \ldots, n$, $a_i + b_i \geq 0$.*

Proof. Since $\text{Rev}(S)$ is submodular, for any subset $S \subseteq \{1, 2, \ldots, i-1\}$

$$\text{Rev}(S \cup \{i\}) - \text{Rev}(S) \geq \text{Rev}(S \cup \{i, \ldots, n\}) - \text{Rev}(S \cup \{i+1, \ldots, n\}).$$

$x^0 = \vec{0}$, $\bar{x}^0 = \vec{1}$;
for $i = 1, \ldots, n$ **do**
  $a_i = \max(0, F(x^{i-1} + e_i) - F(x^{i-1}))$;
  $b_i = \max(0, F(\bar{x}^{i-1} - e_i) - F(\bar{x}^{i-1}))$;
  $p_i = a_i/(a_i + b_i)$ ▷ If $a_i, b_i = 0$, set $p_i = 1$. ;
  $x^i = x^{i-1} + p_i e_i$;
  $\bar{x}^i = \bar{x}^{i-1} - (1 - p_i)e_i$;
end
;
$q^0 = x^n$, $S = \emptyset$;
for $i = 1, \ldots, n$ **do**
  if $F(q^{i-1} + (1 - p_i)e_i) \geq F(q^{i-1} - p_i e_i)$ **then**
    $S = S \cup \{i\}$;
    $q^i = q^i + (1 - p_i)e_i$;
  else
    $q^i = q^i - p_i e_i$;
  end
end
return S ;

Algorithm 2: Submodular algorithm for the assortment optimization problem under the 2-product nonparametric choice model.

Note that by definition of x^{i-1} and $\bar{x}^{i-1}$,

$$\begin{aligned}
a_i = & \sum_{S \subseteq \{1,2,\ldots,i-1\}} \prod_{j \in S} p_j \prod_{j \notin S} (1 - p_j) \mathrm{Rev}(S \cup \{i\}) \\
& - \sum_{S \subseteq \{1,2,\ldots,i-1\}} \prod_{j \in S} p_j \prod_{j \notin S} (1 - p_j) \mathrm{Rev}(S) \\
= & \sum_{S \subseteq \{1,2,\ldots,i-1\}} \prod_{j \in S} p_j \prod_{j \notin S} (1 - p_j) \left[\mathrm{Rev}(S \cup \{i\}) - \mathrm{Rev}(S) \right] \\
\geq & \sum_{S \subseteq \{1,2,\ldots,i-1\}} \prod_{j \in S} p_j \prod_{j \notin S} (1 - p_j) \left[\mathrm{Rev}(S \cup \{i, \ldots, n\}) - \mathrm{Rev}(S \cup \{i+1, \ldots, n\}) \right] \\
= & - b_i,
\end{aligned}$$

which gives the result. □

We now show that this algorithm is indeed a 2-approximation algorithm. Suppose that we can show that at the end of the first stage that $2F(x^n) = 2F(\bar{x}^n) \geq$

OPT. Then, in the second stage, we are simply applying the method of conditional expectations such that for each iteration i

$$\begin{aligned} F(q^i) &= \max(F(q^{i-1} + (1-p_i)e_i), F(q^{i-1} - p_i e_i)) \\ &\geq p_i F(q^{i-1} + (1-p_i)e_i) + (1-p_i)F(q^{i-1} - p_i e_i) \\ &= F(q^{i-1}), \end{aligned}$$

where the first inequality follows from the fact that the maximum of two values is greater than or equal to any convex combination of the two. Applying this iteratively implies that the revenue returned by our algorithm will be

$$2\text{ALG} = 2F(q^n) \geq 2F(q^0) = 2F(x^n) \geq \text{OPT}.$$

Therefore, we just need to focus on bounding $F(x^n)$.

We first present the linear programming relaxation against which we will bound the algorithm. Let y_i be a binary variable representing whether or not we offer product i and z_g be a binary variable representing whether or not a customer of class g will purchase her second choice product g_2 given y. Then, we can solve the assortment optimization problem with the following integer program.

$$\begin{aligned} \text{maximize} \quad & \sum_{g \in \mathcal{G}} \lambda_g \left[r_{g_1} y_{g_1} + r_{g_2} z_g \right] \\ \text{s.t.} \quad & y_{g_1} + z_g \leq 1 \quad \forall g \in \mathcal{G} \\ & z_g \leq y_{g_2} \quad \forall g \in \mathcal{G} \\ & y, z \in \{0,1\} \end{aligned}$$

Let (y^*, z^*) be an optimal solution to the linear programming (LP) relaxation of the above integer program and let OPT_{LP} be the optimal value. Note that since this is a maximization problem, we know $z_g = \min(1 - y_{g_1}, y_{g_2})$ for all $g \in \mathcal{G}$.

We will compare B_i to the revenue of a fractional solution using the optimal LP values. Given a fractional $y \in [0,1]^n$, we define

$$w(y) := \sum_{g \in \mathcal{G}} \lambda_g r_{g_1} y_{g_1} + \lambda_g r_{g_2} \min(y_{g_2}, 1 - y_{g_1}).$$

Suppose that we have already set probabilities $p_1, \ldots, p_i$ in the first stage of our algorithm. Let $y^i = (y_1^i, ..., y_n^i)$ be a random variable in which each component is independent and $y_j^i = 1$ with probability p_j for all $j \leq i$ and $y_j^i = y_j^*$ for all $j > i$. To give some intuition, y^i acts like our current probabilistic solution on $1, \ldots, i$ and like the optimal LP solution on the remaining variables. At the start, $\mathbb{E}[w(y^0)] = w(y^*) = \text{OPT}_{\text{LP}}$, but by the end of the first stage of the algorithm $\mathbb{E}[w(y^n)] = F(x^n)$. Therefore, we will look at bounding how $\mathbb{E}[w(y^i)]$ changes.

Recall that B_i was the average expected revenues between two extremes for the remaining probabilities in which we either offer all products with index i or higher or we do not offer any of these products. We will show that in fact the difference $B_i - B_{i-1}$ can be related to the expected difference $\mathbb{E}[w(y^{i-1}) - w(y^i)]$.

Lemma 4.2.6. *For $i = 1, 2, \ldots, n$,*

$$\mathbb{E}[w(y^{i-1}) - w(y^i)] \leq B_i - B_{i-1}.$$

Proof. Suppose that $y_i^i = 1$. Then, $\mathbb{E}[w(y^{i-1}) - w(y^i) | y_i^i = 1]$ is the difference in the expected fractional revenue when we reduce y_i^i from 1 to y_i^*. Consider any customer class that has $g_1 = i$. As y_i^i decreases, we know that fractional revenue decreases by at least $\lambda_g r_i (1 - y_i^*)$. However, we can also increase the expected fractional revenue by at most $\lambda_g r_{g_2} (1 - y_i^*) \bar{x}_{g_2}^{i-1}$. Note if $g_2 \in \{1, \ldots, i-1\}$ this is the exact increase in expected fractional revenue and if $g_2 > i$ then $\bar{x}_{g_2}^{i-1} = 1$ and this is an upper bound. Now consider any customer class that has $g_2 = i$. Then, by

decreasing y_i^i we decrease the fractional revenue by at least $\lambda_g r_{g_2}(1-y_i^*)(1-\bar{x}_{g_1}^i)$. All other customer classes will not contribute to the difference. Overall,

$$\begin{aligned}
&\mathbb{E}[w(y^{i-1}) - w(y^i)|y_i^i = 1] \\
&\leq (1-y_i^*)\left[-\sum_{g:g_1=i}\lambda_g r_i - \sum_{g:g_2=i}\lambda_g r_i(1-\bar{x}_{g_1}^{i-1}) + \sum_{g:g_1=i}\lambda_g r_{g_2}\bar{x}_{g_2}^{i-1}\right] \\
&= (1-y_i^*)b_i.
\end{aligned}$$

Similarly, suppose that $y_i^i = 0$. Then, $\mathbb{E}[w(y^{i-1}) - w(y^i)|y_i^i = 0]$ is the difference in the expected fractional revenue when we increase y_i^i from 0 to y_i^*. Consider any customer class that has $g_1 = i$. As y_i^i increases, we know that fractional revenue increases by at most $\lambda_g r_i y_i^*$. Furthermore, we decrease the fractional revenue by at least $\lambda_g r_{g_2} y_i^* x_{g_2}^i$. Next consider any customer class that has $g_2 = i$. Then, by increasing y_i^i we can increase the fractional revenue by at most $\lambda_g r_{g_2} y_i^*(1-x_{g_2}^i)$. Overall,

$$\begin{aligned}
\mathbb{E}[w(y^{i-1}) - w(y^i)|y_i^i = 0] &\leq y_i^*\left[\sum_{g:g_1=i}\lambda_g r_i + \sum_{g:g_2=i}\lambda_g r_i(1-x_{g_1}^{i-1}) - \sum_{g:g_1=i}\lambda_g r_{g_2}x_{g_2}^{i-1}\right] \\
&= y_i^* a_i.
\end{aligned}$$

We can now find the overall expected value in the difference.

$$\begin{aligned}
&\mathbb{E}[w(y^{i-1}) - w(y^i)] \\
&= p_i\mathbb{E}[w(y^{i-1}) - w(y^i)|y_i^i = 1] + (1-p_i)\mathbb{E}[w(y^{i-1}) - w(y^i)|y_i^i = 0] \\
&\leq p_i(1-y_i^*)b_i + (1-p_i)y_i^* a_i.
\end{aligned}$$

Recall by Lemma 4.2.5 that $a_i + b_i \geq 0$. If $a_i \leq 0$, then we know $b_i \geq 0$ and we set $p_i = 0$. Thus, $\mathbb{E}[w(y^{i-1}) - w(y^i)] \leq 0$ but $B_i - B_{i-1} = \frac{1}{2}b_i \geq 0$. The same holds if $b_i \leq 0$. Therefore, the last case we have to worry about is when $a_i > 0$

and $b_i > 0$. In this case, $p_i = a_i/(a_i + b_i)$ and

$$\begin{aligned}
\mathbb{E}[w(y^{i-1}) - w(y^i)] &\leq p_i(1 - y_i^*)b_i + (1 - p_i)y_i^* a_i \\
&= \frac{a_i}{a_i + b_i}(1 - y_i^*)b_i + \frac{b_i}{a_i + b_i}y_i^* a_i \\
&= \frac{a_i b_i}{a_i + b_i} \\
&\leq \frac{1}{2} \cdot \frac{a_i^2 + b_i^2}{a_i + b_i} \\
&= B_i - B_{i-1},
\end{aligned}$$

where the last inequality follows from the fact that for all $a, b \in \mathbb{R}$, $a^2 + b^2 \geq 2ab$. □

Theorem 4.2.7. $2F(x^n) \geq \mathrm{OPT}_{\mathrm{LP}}$.

Proof. Summing up the result in Lemma 4.2.6 for $i = 1, \ldots, n$,

$$\mathbb{E}[w(y^0)] - \mathbb{E}[w(y^n)] \leq B_n - B_0.$$

Recall that at the beginning of the algorithm, $\mathbb{E}[w(y^0)] = \mathrm{OPT}_{\mathrm{LP}}$ and at the end of the algorithm $\mathbb{E}[w(y^n)] = F(x^n)$. On the other hand,

$$B_0 = \frac{1}{2}[F(\vec{0}) + F(\vec{1})] = \frac{1}{2}\sum_{g \in \mathcal{G}} \lambda_g r_{g_1} \geq 0$$

and

$$B_n = \frac{1}{2}[F(x^n) + F(\bar{x}^n)] = F(x^n).$$

Thus,

$$\mathrm{OPT}_{\mathrm{LP}} - F(x^n) \leq F(x^n),$$

which gives the result. □

This shows that we can construct a 2-approximation algorithm for this problem that is deterministic and runs in time linear in the input size.

Unfortunately, neither of the algorithms presented so far extends to larger values of k. Therefore, in the next section we present a general, deterministic algorithm that relies upon rounding a linear programming solution.

4.3 LP Algorithm for Longer Preference Lists

In this section, we present a general algorithm for the assortment optimization problem under the k-product nonparametric choice model. We can formulate this optimization problem as an integer program (IP). For each product $i \in N$, let $x_i \in \{0, 1\}$ be a variable representing whether or not we offer product i. Further for each customer class $g \in \mathcal{G}$ and $j \in \{1, 2, \ldots, k\}$, we let g_j be the jth preferred product for customers in class g and let y_{g,g_j} be a variable representing whether or not a customer of type g purchases this product. Then, the assortment optimization problem can be formulated as the following IP.

$$
\begin{aligned}
\text{maximize} \quad & \sum_{g} \sum_{j=1}^{k} \lambda_g r_{g_j} y_{g,g_j} & \\
\text{subject to} \quad & \sum_{j=1}^{k} y_{g,g_j} \leq 1 & \forall g \in \mathcal{G} \\
& y_{g,g_j} \leq x_{g_j} & \forall g \in \mathcal{G}, j \in \{1, 2, \ldots, k\} \\
& \sum_{i=j+1}^{k} y_{g,g_i} \leq 1 - x_{g_j} & \forall g \in \mathcal{G}, j \in \{1, 2, \ldots, k-1\} \\
& x_i \in \{0, 1\} & \forall i \in N \\
& y_{g,g_j} \in \{0, 1\} & \forall g \in \mathcal{G}, j \in \{1, 2, \ldots, k\}
\end{aligned}
$$

The objective function calculates the total expected revenue. The first constraint states that a customer of type g can make at most one purchase, the second

constraint states that a customer can only purchase an offered product, and the third constraint states that if a customer's jth preferred product is available then they cannot purchase a less preferred product. This formulation was previously introduced by Bertsimas and Mišic [6]. The linear programming (LP) relaxation of this IP is given by

$$\begin{aligned}
\text{maximize} \quad & \sum_{g} \sum_{j=1}^{k} \lambda_g r_{g_j} y_{g,g_j} & \\
\text{subject to} \quad & \sum_{j=1}^{k} y_{g,g_j} \leq 1 & \forall g \in \mathcal{G} \\
& y_{g,g_j} \leq x_{g_j} & \forall g \in \mathcal{G}, j \in \{1,2,\ldots,k\} \\
& \sum_{i=j+1}^{k} y_{g,g_i} \leq 1 - x_{g_i} & \forall g \in \mathcal{G}, j \in \{1,2,\ldots,k-1\} \\
& x_i \in [0,1] & \forall i \in N \\
& y_{g,g_j} \in [0,1] & \forall g \in \mathcal{G}, j \in \{1,2,\ldots,k\}
\end{aligned}$$

where the integer variable constraints have been replaced to allow fractional solutions.

We will use the LP relaxation of this IP to inform our decisions about which products to offer. Let (x^*, y^*) be an optimal LP solution with value z^*. From this fractional solution, we will construct an assortment $\bar{x}$. For any product i such that $x_i^* \in \{0,1\}$, we set $\bar{x}_i = x_i^*$. On the other hand, if $0 < x_i^* < 1$, then we offer product i independently with probability $\frac{1}{2k} + \frac{1}{k} x_i^*$. By doing so, we strike a balance between the LP solution and a random assortment. Note that the probability of offering a product decreases as k increases.

To complete the analysis of this algorithm, we will compare the expected revenue of $\bar{x}$ for a single customer class g and product g_j compared to the LP value y_{g,g_j}^* placed on this product.

Lemma 4.3.1. *For a customer class $g \in \mathcal{G}$ and $j \in \{1, 2, \ldots, k\}$,*

$$\mathbb{E}[\lambda_g r_{g_j}(1-\bar{x}_{g_1})(1-\bar{x}_{g_2})\ldots(1-\bar{x}_{g_{j-1}})\bar{x}_{g_j}] \geq \beta_{j,k}\lambda_g r_{g_j} y^*_{g,g_j},$$

where

$$\beta_{j,k} = \min_{0\leq y\leq 1} \frac{1}{y}\left(1-\frac{3}{2k}+\frac{y}{k}\right)^{j-1}\left(\frac{1}{2k}+\frac{y}{k}\right).$$

Proof. If $x^*_{g_i} = 1$ for any $i < j$, then the algorithm always offers product g_i and does not gain any revenue from g_j. Similarly, by the third constraint in the LP, we have that $y^*_{g,g_j} = 0$ and the LP does not gain any revenue from this product either. Similarly, if $x^*_{g_j} = 0$, then algorithm does not gain any revenue from this product and, by the second constraint in the LP, we have that $y^*_{g,g_j} = 0$ and so neither does the LP.

In the remaining cases, we offer product g_i with probability 0 if $x^*_{g_i} = 0$ and probability $\frac{1}{2k} + \frac{1}{k}x^*_{g_i}$ if $0 < x^*_{g_i} < 1$, for all $i < j$. Additionally, we offer product g_j with probability 1 if $x^*_{g_j} = 1$ and probability $\frac{1}{2k} + \frac{1}{k}x^*_{g_j}$ if $0 < x^*_{g_j} < 1$ otherwise. This yields expected revenue

$$\begin{aligned}
&\mathbb{E}[\lambda_g r_{g_j}(1-\bar{x}_{g_1})(1-\bar{x}_{g_2})\ldots(1-\bar{x}_{g_{j-1}})\bar{x}_{g_j}] \\
&\geq \lambda_g r_{g_j}\left(1-\frac{1}{2k}-\frac{x^*_{g_1}}{k}\right)\left(1-\frac{1}{2k}-\frac{x^*_{g_2}}{k}\right)\ldots\left(1-\frac{1}{2k}-\frac{x^*_{g_{j-1}}}{k}\right)\left(\frac{1}{2k}+\frac{x^*_{g_j}}{k}\right) \\
&= \lambda_g r_{g_j}\left(1-\frac{3}{2k}+\frac{1-x^*_{g_1}}{k}\right)\left(1-\frac{3}{2k}+\frac{1-x^*_{g_2}}{k}\right)\ldots\left(1-\frac{3}{2k}+\frac{1-x^*_{g_{j-1}}}{k}\right)\left(\frac{1}{2k}+\frac{x^*_{g_j}}{k}\right) \\
&\geq \lambda_g r_{g_j}\left(1-\frac{3}{2k}+\frac{y*_{g,g_j}}{k}\right)^{j-1}\left(\frac{1}{2k}+\frac{y^*_{g,g_j}}{k}\right) \\
&\geq \beta_{j,k}\lambda_g r_{g_j} y^*_{g,g_j},
\end{aligned}$$

where the second to last inequality comes from the second and third constraints of the LP and the last inequality comes from the definition of $\beta_{j,k}$. □

Theorem 4.3.2. *We can efficiently find an assortment with revenue at least*

$$\left(1-\frac{1}{k}\right)^{k-1}\frac{2}{k}\cdot \mathrm{OPT}.$$

Thus, there exists a 2-approximation algorithm for the assortment optimization under the 2-product nonparametric choice model, a 3.375-approximation algorithm under the 3-product nonparametric choice model, and a 4.741-approximation algorithm under the 4-product nonparametric choice model.

Proof. Lemma 4.3.1 shows that the proposed LP rounding algorithm is a $\beta_{k,k}$-approximation algorithm. Further, this algorithm can be derandomized using the method of conditional expectations in polynomial time. It remains to analyze the value of $\beta_{k,k}$ where

$$\beta_{k,k}=\min_{0\le y\le 1}\left(1-\frac{3}{2k}+\frac{y}{k}\right)^{k-1}\left(\frac{1}{2ky}+\frac{1}{k}\right).$$

The derivative of the inner minimization is given by

$$\frac{k-1}{k}\left(1-\frac{3}{2k}+\frac{y}{k}\right)^{k-2}\left(\frac{1}{2ky}+\frac{1}{k}\right)-\frac{1}{2ky^2}\left(1-\frac{3}{2k}+\frac{y}{k}\right)^{k-1}.$$

Setting this expression equal to zero and simplifying we obtain the following expression

$$\frac{k-1}{k}\left(\frac{1}{2ky}+\frac{1}{k}\right)-\frac{1}{2ky^2}\left(1-\frac{3}{2k}+\frac{y}{k}\right)=0.$$

Multiplying by $2k^2y^2$,

$$2(k-1)y^2+(k-2)y-k+\frac{3}{2}=0.$$

This quadratic has roots $\frac{1}{2}$ and $-\frac{k-1/2}{k-1}$. Since the latter is negative, we can easily see that the function is minimized at $y=\frac{1}{2}$ and

$$\beta_{k,k}=\left(1-\frac{1}{k}\right)^{k-1}\frac{2}{k},$$

which gives the result. □

It is clear that the limiting factor for the randomized LP rounding algorithm proposed above is the revenue derived from the last choice product of each customer class. In Section 4.5, we show that this LP rounding algorithm performs near optimal in practice. Further, the theoretical performance improves upon the ratio of Aouad et al. [3] for all k.

Lastly, we present some interesting structure for the rounding algorithm. In the lemma below we show that a modified version of the linear program is half-integral meaning that $x_i^* \in \{0, \frac{1}{2}, 1\}$ for all $i = 1, \ldots, n$. This implies that our rounding scheme is actually offering each product with probability in $\{0, 1/k, 1\}$. Further, our previous analysis for the approximation ratio holds for this relaxation as well. First, we present the following integer program. Interestingly, when $k = 2$ this is equivalent to the full integer program.

$$
\begin{aligned}
\text{maximize} \quad & \sum_{g} \sum_{j=1}^{k} \lambda_g r_{g_1} y_{g,g_j} & \\
\text{subject to} \quad & y_{g,g_j} \leq x_{g,g_j} & \forall g \in \mathcal{G}, 1 \leq j \leq k\} \\
& y_{g,g_j} \leq 1 - x_{g_i} & \forall g \in \mathcal{G}, 1 \leq i < j \leq k \\
& x_i \in \{0,1\} & \forall i \in N \\
& y_{g,g_j} \in \{0,1\} & \forall g \in \mathcal{G}, j \in \{1,2,\ldots,k\}
\end{aligned}
$$

Given the form of this integer program, which contains at most two variables in each inequality, Hochbaum et al. [29] show that there exists an optimal half-integral solution to the linear programming relaxation. We additionally show that all basic feasible solutions are half-integral. The proof will follow a similar argument to that by Nemhauser and Trotter [48], who show that basic feasible solutions for the vertex cover problem LP are half-integral.

Lemma 4.3.3. *Let (x^*, y^*) be any optimal basic feasible solution to the modified*

LP. Then this solution is half-integral in x^*, i.e. $x_i^* \in \{0, \frac{1}{2}, 1\}$ for all $i = 1, \ldots, n$, which implies that y^* is also half-integral.

Proof. Assume by way of contradiction that (x^*, y^*) is not half-integral. We will show that (x^*, y^*) is not a basic solution. First, for any $\varepsilon > 0$, let

$$\bar{x}_i = \begin{cases} x_i^* + \varepsilon & 0 < x_i^* < \frac{1}{2}, \\ x_i^* - \varepsilon & \frac{1}{2} < x_i^* < 1, \\ x_i^* & \text{otherwise.} \end{cases} \quad x_i' = \begin{cases} x_i^* - \varepsilon & 0 < x_i^* < \frac{1}{2}, \\ x_i^* + \varepsilon & \frac{1}{2} < x_i^* < 1, \\ x_i^* & \text{otherwise.} \end{cases}$$

Furthermore, for all $g \in \mathcal{G}$ and $j \in \{1, 2, \ldots, k\}$,

$$\bar{y}_{g,g_j} = \begin{cases} y_{g,g_j}^* + \varepsilon & 0 < y_{g,g_j}^* < \frac{1}{2}, \\ y_{g,g_j}^* - \varepsilon & \frac{1}{2} < y_{g,g_j}^* < 1, \\ y_{g,g_j}^* & \text{otherwise.} \end{cases} \quad y_{g,g_j}' = \begin{cases} y_{g,g_j}^* - \varepsilon & 0 < y_{g,g_j}^* < \frac{1}{2}, \\ y_{g,g_j}^* + \varepsilon & \frac{1}{2} < y_{g,g_j}^* < 1, \\ y_{g,g_j}^* & \text{otherwise.} \end{cases}$$

Clearly, $x^* = \frac{1}{2}\bar{x} + \frac{1}{2}x'$ and $y^* = \frac{1}{2}\bar{y} + \frac{1}{2}y'$. Therefore, we just need to show both $(\bar{x}, \bar{y})$ and (x', y') are feasible solutions to the LP for small enough ε to show that (x, y) cannot be a basic feasible solution.

We start with $(\bar{x}, \bar{y})$. Note that by setting ε small enough we maintain that $0 \leq \bar{x} \leq 1$ and $0 \leq \bar{y} \leq 1$. It remains to show the feasibility of the two LP inequalities. Consider the first inequality for $g \in \mathcal{G}$ and $1 \leq j \leq k$. Note that if $y_{g,g_j}^* < x_{g_j}^*$, then for small enough ε this constraint will continue to hold for $(\bar{x}, \bar{y})$. On the other hand, if $y_{g,g_j}^* = \bar{x}_{g_j}^*$ then either neither LP value is adjusted or y_{g,g_j}^* is adjusted in the same direction as $x_{g_j}^*$. Again, this implies that the constraint continues to hold for $(\bar{x}, \bar{y})$. Similarly, consider the second LP inequality for $g \in \mathcal{G}$ and $1 \leq i < j \leq k$. If $y_{g,g_j}^* < 1 - x_{g_i}^*$, then for small enough ε this constraint will continue to hold for $(\bar{x}, \bar{y})$. On the other hand, if $y_{g,g_j}^* = 1 - x_{g_i}^*$, then either

neither LP value is adjusted or y^*_{g,g_j} is adjusted in the opposite direction as $x^*_{g_i}$ and the constraint will continue to hold. Thus, for small enough ε, $(\bar{x}, \bar{y})$ is feasible. A similar argument can be made for (x', y'). □

4.4 Adding a Cardinality Constraint for the 2-Product Choice Model

In this section, we are able to extend the ideas in Section 4.3 to develop an LP rounding algorithm for the cardinality constrained assortment problem under the 2-product nonparametric choice model. In this version of the problem, the firm chooses $S \subseteq N$ to maximize expected revenue with the added constraint that $|S| \leq C$ for some fixed integer C. An integer program for this problem under the k-product nonparametric choice model is given by

$$
\begin{aligned}
\text{maximize} \quad & \sum_{g} \sum_{j=1}^{k} \lambda_g r_{g_j} y_{g,g_j} & \\
\text{subject to} \quad & \sum_{j=1}^{k} y_{g,g_j} = 1 & \forall g \in \mathcal{G} \\
& y_{g,g_j} \leq x_{g_j} & \forall g \in \mathcal{G}, j \in \{1, 2, \ldots, k\} \\
& \sum_{i=j+1}^{k} y_{g,g_i} \leq 1 - x_{g_i} & \forall g \in \mathcal{G}, j \in \{1, 2, \ldots, k-1\} \\
& \sum_{i=1}^{n} x_i \leq C & \\
& x_i \in \{0, 1\} & \forall i \in N \\
& y_{g,g_j} \in \{0, 1\} & \forall g \in \mathcal{G}, j \in \{1, 2, \ldots, k\}
\end{aligned}
$$

This it the same IP as in Section 4.3 with the added constraint that we can offer at most C products.

As before, we will use the LP relaxation of this IP to inform our decisions on which products to offer. However, we have to be careful when rounding the LP solution that we do not go over the capacity C. Let (x^*, y^*) be an optimal LP solution with value z^*. We construct a corresponding random assortment $\bar{x}$ that offers each product i independently with probability $\frac{1}{2}x_i^*$. Note that this is akin to removing the constant term from the offer probabilities for the unconstrained assortment optimization problem while keeping the term $\frac{1}{2}x_i^*$ to ensure that we still place high enough probability on second choice probabilities.

Lemma 4.4.1.

$$\mathbb{E}[\text{Rev}(\bar{x})] \geq \frac{1}{4} \cdot \text{OPT}.$$

Proof. We analyze the expected revenue of $\bar{x}$ by looking at the expected revenue for each customer class g. Consider a single customer class $g \in \mathcal{G}$. We first consider the expected revenue generated from the first choice product g_1. We offer product g_1 with probability $\frac{1}{2}x_{g_1}^*$. The associated expected revenue is

$$\mathbb{E}[\lambda_g r_{g_1} \bar{x}_{g_1}] = \lambda_g r_{g_1} \left[\frac{1}{2}x_{g_1}^*\right] \geq \frac{1}{2}\lambda_g r_{g_1} y_{g,g_1}^*,$$

where the inequality comes from the second constraint of the LP. Hence the expected revenue from this product is at least $\frac{1}{2}$ that from the LP.

Now consider the expected revenue from the second choice product g_2 for customer class g. We offer product g_1 with probability $\frac{1}{2}x_{g_1}^*$ and offer product g_2 with

probability $\frac{1}{2}x^*_{g_2}$. This yields expected revenue

$$\begin{aligned}
\mathbb{E}[\lambda_g r_{g_2}(1-\bar{x}_{g_1})\bar{x}_{g_2}] &= \lambda_g r_{g_2}\left(1-\frac{x^*_{g_1}}{2}\right)\left(\frac{x^*_{g_2}}{2}\right) \\
&= \lambda_g r_{g_2}\left(\frac{1}{2}+\frac{1-x^*_{g_1}}{2}\right)\left(\frac{x^*_{g_2}}{2}\right) \\
&\geq \lambda_g r_{g_2}\left(\frac{1}{2}+\frac{y^*_{g,g_2}}{2}\right)\left(\frac{y^*_{g,g_2}}{2}\right) \\
&\geq \frac{1}{4}\cdot\lambda_g r_{g_2} y^*_{g,g_2}
\end{aligned}$$

where the second to last inequality comes from the second and third constraints of the LP. In comparison, the LP gains revenue $\lambda_g r_{g_2} y^*_{g,g_2}$. Thus, the expected revenue from the LP rounding algorithm is

$$\begin{aligned}
\mathbb{E}[\sum_g \lambda_g[r_{g_1}\bar{x}_{g_1} + r_{g_2}(1-\bar{x}_{g_1})\bar{x}_{g_2}] &\geq \sum_g \frac{1}{2}\lambda_g r_{g_1} y^*_{g,g_1} + \frac{1}{4}\lambda_g r_{g_2} y^*_{g,g_2} \\
&\geq \frac{1}{4}\cdot z^* \\
&\geq \frac{1}{4}\cdot \mathrm{OPT}.
\end{aligned}$$

This shows that the expected revenue of $\bar{x}$ is at least $1/4$ of the optimal revenue. □

The lemma above shows that the random assortment $\bar{x}$ obtains at least $1/4$ of the optimal revenue in expectation. Further, the expected size of $\bar{x}$ is $\frac{1}{2}\sum_{i=1}^n x^*_i \leq C/2 \leq C-1$. Here, we assume that $C > 2$ since otherwise we can solve this problem trivially by enumeration. However, we cannot just hope to derandomize the algorithm as before using the method of conditional expectations without possibly violating the cardinality constraint. Instead, we will look at rounding pairs of products to transform the solution into an integral assortment.

Lemma 4.4.2. *In polynomial time, we can find an integral assortment x such that*

$$\mathrm{Rev}(x) \geq \mathbb{E}[\mathrm{Rev}(\bar{x})]$$

and

$$\sum_{i=1}^{n} x_i \leq C.$$

Proof. Slightly abusing notation let $x \in [0,1]^n$ be the probability that product i is offered. Then, the expected revenue of the corresponding random assortment is given by the multilinear extension

$$F(x) = \sum_{g} \lambda_g r_{g_1} x_{g_1} + \lambda_g r_{g_2}(1 - x_{g_1}) x_{g_2}$$

and the expected size of the assortment is

$$\sum_{i=1}^{n} x_i.$$

To start, set x_i to be the probability i is offered in $\bar{x}$. Then, the expected size of the assortment is at most $C - 1$. While x is not an integral assortment, choose two products i and j such that x_i and x_j are both fractional. We account for the scenario in which there is only one fractional variable remaining at the end of the proof. Consider the following modified solution for a carefully chosen ε: $x_\varepsilon = x + \varepsilon(e_i - e_j)$, where $e_l \in \mathbb{R}^n$ is a vector of all zeros except for a one in the lth position. If $\varepsilon > 0$, this new solution adds ε to x_i and subtracts ε from x_j and hence the expected size of the assortment is unchanged. If $\varepsilon < 0$, then this solution subtracts ε from x_i and adds ε to x_j. Again, the expected size of the assortment is unchanged.

We now analyze the change $F(x_\varepsilon) - F(x)$ resulting from moving to this new solution. For any customer class $g \in \mathcal{G}$ and product $g_1 \in \{i, j\}$, the change in $\lambda_g r_{g_1} x_{g_1}$ is $+\lambda_g r_{g_1} \varepsilon$ if $g_1 = i$ and $-\lambda_g r_{g_1}$ if $g_1 = j$. In either case, the change is linear in ε. Now consider the change in $\lambda_g r_{g_2}(1 - x_{g_1}) x_{g_2}$, the expected revenue from the second choice product. If $g_2 \in \{i, j\}$ but $g_1 \notin \{i, j\}$, the change is either $+\lambda_g r_{g_2}(1 - x_{g_1})\varepsilon$ or $-\lambda_g r_{g_2}(1 - x_{g_1})\varepsilon$, and if $g_1 \in \{i, j\}$ but $g_2 \notin \{i, j\}$, the change

is either $+\lambda_g r_{g_2} x_{g_2}\varepsilon$ or $-\lambda_g r_{g_2} x_{g_2}\varepsilon$. In either case, the change is linear in ε. Lastly, if $g_1 = i$ and $g_2 = j$, the change is

$$\lambda_g r_{g_j}[-(1 - x_i)\varepsilon - x_j\varepsilon + \varepsilon^2]$$

and if $g_1 = j$ and $g_2 = i$, the change is

$$\lambda_g r_{g_j}[(1 - x_i)\varepsilon + x_j\varepsilon + \varepsilon^2].$$

Thus, $F(x_\varepsilon) - F(x)$ is a quadratic function in ε in which the quadratic coefficient is positive and the constant term is zero. If the linear coefficient of ε is positive, we set $\varepsilon = \min\{1-x_i, x_j\}$. By the equation above, this only increases the expected revenue and either x_i or x_j becomes integer valued due to our choice of ε. Otherwise, if the linear coefficient of ε is negative, we set $\varepsilon = -\min\{1-x_j, x_i\}$. Again, this can only increase the expected revenue and at least one of the two variable becomes integer. Further, we have not increased the overall expected size of the assortment.

Thus, after at most n iterations there is at most one fractional value x_i. As before, either setting $x_i = 0$ or $x_i = 1$ improves the expected revenue. In the end, we have transformed x into an integer solution of size $\leq C$ while only increasing the expected revenue. □

Combining Lemma 4.4.1 and Lemma 4.4.2 yields the following theorem.

Theorem 4.4.3. *There exists a 4-approximation algorithm for the cardinality constrained assortment optimization problem under the 2-Product Nonparametric choice model.*

4.5 Computational Experiments

In this section, we provide computational experiments which demonstrate the effectiveness of the LP-based rounding procedure from Section 4.3 used to solve the assortment optimization problem under the k-product nonparametric choice model. We generate a series of random test cases for $k = 3, 4$ and compare the performance of the assortments produced by our algorithm with those produced by the randomized algorithm of Aouad et al. [3]. Not only does our algorithm perform significantly better than this benchmark, but its performance is near optimal in the majority of test cases.

We generate each test instance using the following procedure. First, we set the maximum length of any preference list to be $k \in \{3, 4\}$, the number of products to be $n \in \{50, 100\}$, and the number of customer classes to be $m \in \{1000, 5000, 10000, 50000\}$. Next, we sample the revenues of each product independently from a uniform distribution over $[1, 100]$. We choose the collection of m preference lists by uniformly sampling from the list of all possible preference lists for the given value of k. In other words, for fixed $k \in \{3, 4\}$ and $n \in \{50, 100\}$, we generate all possible preference lists of length $\leq k$. From this set, we uniformly sample without replacement m preference lists. Lastly, we generate the arrival probabilities for each customer class by first generating $(\beta_1, \ldots, \beta_m)$ independently from a uniform distribution over $[0, 1]$ and then setting $\lambda_g = \beta_g / \sum_{k=1}^{m} \beta_k$.

The variation of $(k, n, m) \in \{3, 4\} \times \{50, 100\} \times \{1000, 5000, 10000\}$ gives 12 parameter combinations. For each set of parameters, we generate 100 independent problem instances using the approach outlined above. For each problem instance, we employ two different approaches. The first is the randomized linear program rounding algorithm in Section 4.3, which we refer to as ALG. The second approach

is the randomized algorithm of Aouad et al. [3], which randomly offers each product with probability $1/k$. We refer to this algorithm as RAND. To check the quality of the proposed assortments given by the two approaches, we find the gap between the expected revenue of these assortments and the upper bound provided by the linear programming formulation in Section 4.3.

4.5.1 Results

Our results are given in Table 4.1. The first three columns give the parameter combinations that define each test case. Recall that for each test case, we generate 100 problem instances. Let ALG^p and RAND^p be the expected revenues of the assortments produced by ALG and RAND on problem instance p. Further, let LP^p by the objective of the LP relaxation of the assortment problem for problem instance p, which is trivially an upper bound on the optimal expected revenue. Then, the percent optimality gaps for the two algorithms are given by $100 \times (\text{LP}^p - \text{ALG}^p)/\text{LP}^p$ and $100 \times (\text{LP}^p - \text{RAND}^p)/\text{LP}^p$, respectively. Column 4 in Tables 4.1a and 4.1b reports the average optimality gap over all 100 problem instances, column 5 reports the gap at the 75th percentile, and column 6 reports the largest gap seen.

The results in Table 4.1 clearly indicate that the randomized LP rounding algorithm far outperforms the randomized algorithm of Aouad et al. [3] on all metrics. Over all parameter combinations, the average performance of ALG is never below 0.52% of optimal. Further, over all 1200 problem instances, the percent optimality gap of ALG never exceeds 3.7%. On the other hand, the average percent optimality gap of RAND is never below 2.9% and this algorithm suffers from optimality gaps of over 10% for certain problem instances.

			% Optimality Gap		
k	n	m	Avg.	75th	Max.
3	50	1000	0.11	0.0	0.86
3	50	5000	0.01	0.0	1.25
3	50	10000	0.05	0.0	1.01
3	100	1000	0.06	0.0	0.59
3	100	5000	0.05	0.0	0.98
3	100	10000	0.05	0.0	0.51
4	50	1000	0.37	0.0	2.62
4	50	5000	0.41	0.0	2.60
4	50	10000	0.52	0.0	3.66
4	100	1000	0.22	0.0	2.42
4	100	5000	0.32	0.0	1.94
4	100	10000	0.33	0.0	3.29

(a) Performance of ALG on 1200 randomly generated instances.

			% Optimality Gap		
k	n	m	Avg.	75th	Max.
3	50	1000	3.29	2.07	8.34
3	50	5000	2.93	1.97	6.45
3	50	10000	3.08	2.04	6.85
3	100	1000	3.51	2.88	6.00
3	100	5000	2.91	2.35	5.73
3	100	10000	2.90	2.20	5.45
4	50	1000	4.97	3.51	9.93
4	50	5000	5.06	3.75	10.37
4	50	10000	4.88	3.87	8.94
4	100	1000	4.95	4.04	8.79
4	100	5000	4.74	4.17	8.25
4	100	10000	4.45	3.45	8.45

(b) Performance of RAND on 1200 randomly generated instances.

Table 4.1: Comparing the performance of the two randomized algorithms.

As k grows, substitution behavior becomes more complicated, and as a result, we would expect the performance of both algorithms to deteriorate. This trend is precisely what we observe; for ALG the average percent optimality gap grows from 0.055% to 0.36% as k goes from 3 to 4 and the percent gap for RAND also grows from to 3.10% to 4.84%. Surprisingly, we observe that both algorithms perform worst for smaller problem instances. In fact, the worst optimality gaps for both algorithms are when $n = 50$ and $m = 1000$. It might be that in these smaller instances there is a smaller margin for error and hence there are fewer assortments that have expected revenue close to optimal.

4.6 Estimation and Analysis

In this section, we measure the marginal benefit of fitting nonparametric choice models with preference lists of increasing lengths. Specifically, we fit k-product

nonparametric choice models for $k \in \{1, 2, 3, 4, 5\}$ and then assess the efficacy of these fitted models by measuring both how accurately they predict purchase probabilities and the profitability of the assortments they recommend. More specifically, the first metric that we use to assess the fitted models is the mean absolute error (MAE) of the predicted purchase probabilities over all possible assortments. To compute the second metric, we use the generated fit to find the optimal assortments under the assumption that customer choice is governed by the fitted model. We then check the expected revenue of these assortments in practice. Both metrics require us to know the true model, referred to as the ground choice model, that governs how customers make purchases. We assume that this ground choice model comes in the form of a nonparametric choice model whose preference lists are generated from the survey data collected in Kamishima [37]. We use these preference lists, along with randomly generated arrival probabilities, to simulate customers purchasing sushi from a restaurant.

In Kamishima [37], the authors asked 5000 unique customers to rank 10 popular sushi items from most preferred to least preferred. We use these 5000 rankings as the basis for the preference lists in the ground choice model. The 10 popular sushi items that were ranked were Shrimp, Sea Eel, Tuna, Squid, Sea Urchin, Salmon Roe, Egg, Fatty Tuna, Tuna Roll, and Cucumber Roll. We let N denote this collection of sushi items, which we assume are the only sushi items ever available for purchase. Further, we let $n = |N| = 10$.

Instead of using the full rankings of each surveyed customer as the preference lists, we randomly truncate each ranking to create a preference list. There are many reasons why we choose to truncate the full rankings. First, if each preference list is allowed to be the full ranked list, then every customer will always make a purchase.

This implies that the optimal assortment under this choice model will simply be the sushi item with the highest revenue, since this item will appear in every customer's preference list. The ultimate goal of any retailer using customer choice models is to use these fitted models to uncover valuable assortments, which can then be offered to increase profits. By truncating the preference lists, we create a setting in which it is far more difficult to discover profitable assortments and hence the profitability of the recommended assortment becomes a useful metric for comparing various models. Finally, by truncating the preference lists, we work under the assumption that customers are not willing to substitute more than a handful of times beyond their most preferred product; very rarely is a customer willing to settle for her ninth or tenth most preferred product.

4.6.1 Experimental Setup

In the ground truth choice model, we let the set of customer types be given by $\mathcal{G} = \{1, \ldots, 5000\}$. The full ranking of length 10 and the arrival probability of each customer class $g \in \mathcal{G}$ are respectively given by σ_g and λ_g. To generate the truncated preference list of each customer class, we sample a cut-off u_g uniformly from $\{1, \ldots, \mathcal{U}\}$ for $\mathcal{U} \in \{4, 6, 8\}$ and set $\sigma'_g = [\sigma_g(1), \ldots, \sigma_g(u_g)]$. By varying $\mathcal{U}$, we capture a reasonable breadth of substitution behavior. Last, we generate the arrival probabilities for each customer class by first generating $(\beta_1, \ldots, \beta_{5000})$ independently from a uniform distribution over $[0, 1]$ and then setting $\lambda_g = \beta_g / \sum_{i=1}^{5000} \beta_i$.

Once the ground choice model has been generated, we then generate the historical sales data under the assumption that the purchasing behavior of all arriving customers is governed by the ground choice model. We assume that we have access to the past purchasing history of τ customers. We vary $\tau \in \{1000, 5000, 10000\}$

to test each model's robustness to data availability. We assume that the historical purchase history is captured by $\text{PH}_\tau = \{(S_t, z_t) : t = 1, \ldots, \tau\}$, where S_t is the subset of products offered to customer in time period t and z_t is the product purchased in time period t. If the customer leaves without making a purchase, we set $z_t = 0$. We form the subset S_t in each time period by randomly including each product with probability 0.5. In this way, we uniformly sample an assortment in each time period. The class g_t that customer t belongs to is sampled from the distribution $(\lambda_1, \ldots, \lambda_{5000})$. Given that the ground truth model is a nonparametric choice model, we set $z_t = \text{argmin}_{i \in S_t} \sigma'_{g_t}(i)$. We generate the data such that $\text{PH}_{1000} \subset \text{PH}_{5000} \subset \text{PH}_{10000}$. This limits the role that randomness plays in confounding the affect of additional data availability.

For each choice of $\mathcal{U}$, we generate 10 distinct ground choice models. Then, for each of these choice models we generate PH_{10000} using the sales data generation process described above. Lastly, for each of these past purchase histories, we generate 100 different revenues for the products, where the revenue of each product is chosen uniformly from the interval $[0, 100]$. This gives us $3 \times 3 \times 10 \times 100 = 9000$ datasets where dataset DS_j is associated with a purchase history PH_τ^j and a set of revenues $(r_1^j, \ldots, r_{10}^j)$. For each dataset DS_j generated from ground choice model GC, we fit nonparametric choice models $\text{NP}_k \ \forall \ k \in \{1, 2, \ldots, 5\}$, where NP_k is a k-product nonparametric choice model, and we also fit a multinomial logit model ML, which we use as a benchmark. For each of the six models, we use maximum likelihood estimation (MLE) to find the optimal set of parameters of the corresponding model. It is well documented (see Train [57] for MNL and van Ryzin and Vulcano [58] for the nonparametric model) that the log-likelihood functions under both the nonparametric choice model and the MNL model are concave and thus MLE is tractable.

Next, we describe how we measure the accuracy of the fitted models. Suppose for dataset DS_j generated from ground choice model GC, we have fitted choice models $\{\mathrm{NP}_1, \ldots, \mathrm{NP}_5, \mathrm{ML}\}$. For choice model $\mathrm{CM} \in \{\mathrm{NP}_1, \ldots, \mathrm{NP}_5, \mathrm{ML}\}$ let $\mathrm{Pr}_i^{\mathrm{CM}}(S)$ be the probability that product i is purchased under choice model CM when assortment S is offered. We measure the efficacy of the fitted models using two metrics. The first metric is the MAE of the predicted purchase probabilities of each choice model. For fitted choice model CM, we let $\mathrm{MAE(CM)} = \frac{1}{2^n} \sum_{S \subseteq N} \sum_{i \in S} |\mathrm{Pr}_i^{\mathrm{CM}}(S) - \mathrm{Pr}_i^{\mathrm{GC}}(S)|/|S|$ be the average error in predicted purchase probability over every product and every assortment. The second metric considers the profitability of the recommended assortments of each fitted model. For fitted choice model CM, we compute the optimal recommended assortment as $S^j(\mathrm{CM}) = \mathrm{argmax}_{S \subseteq N} \sum_{i \in N} r_i^j \mathrm{Pr}_i^{\mathrm{CM}}(S)$. We then check the performance of these assortments by computing how well these assortments perform under the ground truth choice model. In particular, we compute expected revenues $R^j(\mathrm{CM}) = \sum_{i \in N} r_i^j \mathrm{Pr}_i^{\mathrm{GC}}(S^j(\mathrm{CM}))$ in addition to the optimal expected revenue R^{j*} for dataset DS_j, which we can compute through complete enumeration since we only have 10 products. For each fitted model $\mathrm{CM} \in \{\mathrm{NP}_1, \ldots, \mathrm{NP}_5, \mathrm{ML}\}$, we store the percent optimality gap as computed by $100 \times ((R^{j*} - R^j(\mathrm{CM}))/R^{j*})$.

4.6.2 Results

Tables 4.2a and 4.2b give the performance of the six fitted models across the two metrics described in the previous section for all combinations of $\mathcal{U}$ and τ. The first column in both tables gives the value of $\mathcal{U}$ that was used to generate the ground choice models. Recall that for each value of $\mathcal{U}$, we generate 10 different ground choice models which are then each used to generate a distinct purchase history

$\mathcal{U}$	Model	τ 1000	5000	10000
4	NP_1	17.78	17.78	17.78
4	NP_2	2.85	1.86	1.77
4	NP_3	2.26	1.01	0.68
4	NP_4	2.55	1.23	0.84
4	NP_5	2.89	1.36	0.92
4	ML	3.77	3.71	3.69
6	NP_1	22.13	22.13	22.13
6	NP_2	6.22	5.26	4.89
6	NP_3	2.82	1.77	1.42
6	NP_4	2.20	1.00	0.678
6	NP_5	2.20	0.958	0.725
6	ML	2.72	2.41	2.41
8	NP_1	23.68	23.68	23.68
8	NP_2	8.03	7.42	7.31
8	NP_3	3.56	2.50	2.37
8	NP_4	2.33	1.18	0.892
8	NP_5	1.99	1.18	0.655
8	ML	1.51	1.50	1.51

(a) Average percent optimality gap of revenues of the recommended assortments under the fitted models.

$\mathcal{U}$	Model	τ 1000	5000	10000
4	NP_1	0.0974	0.0974	0.0974
4	NP_2	0.0276	0.0202	0.0185
4	NP_3	0.0250	0.0146	0.0113
4	NP_4	0.0267	0.0153	0.0120
4	NP_5	0.0280	0.0162	0.0128
4	ML	0.0278	0.0255	0.0251
6	NP_1	0.129	0.128	0.128
6	NP_2	0.0495	0.0443	0.0433
6	NP_3	0.0306	0.0208	0.0182
6	NP_4	0.0293	0.0170	0.0138
6	NP_5	0.0301	0.0173	0.0138
6	ML	0.0256	0.0221	0.0216
8	NP_1	0.146	0.145	0.145
8	NP_2	0.0637	0.0602	0.0590
8	NP_3	0.0359	0.0282	0.0255
8	NP_4	0.0315	0.0211	0.0172
8	NP_5	0.0304	0.0188	0.0149
8	ML	0.0224	0.0202	0.0198

(b) Average MAE of the fitted models.

Table 4.2: Measuring the efficacy of the fitted models.

PH_{10000}. For each of these 10 streams of purchase histories, we fit nonparametric choice models NP_k for $k \in \{1, 2, \ldots, 5\}$ and an MNL model ML. The second column in Tables 4.2a and 4.2b denotes the fitted model and columns 3-5 report the performance of these fitted model for each value of $\tau \in \{1000, 5000, 10000\}$. In Table 4.2a, these three columns give the average percent optimality gap of the assortments recommended by each of the fitted models averaged over the 10 purchase histories and the 100 different revenue vectors. In Table 4.2b, columns 3-5 give the average MAE over the 10 purchase histories.

Similarly, in Table 4.3a columns 3-5 report the computation time needed to perform the MLE for each value of τ averaged over the 10 distinct purchase histories.

We report this data so we can better assess the marginal gains from fitting increasingly rich choice models; more complex nonparametric choice models capture a broader set of purchasing behavior but require more time to estimate. Lastly, in Table 4.3b columns 3-5 report the number of customer classes with a nonzero arrival probability given by the MLE for each value of τ averaged over the 10 purchase histories. This number helps us understand, to some degree, the extent to which extra customer classes in the more complex nonparametric models are used in improving the likelihood.

It is not surprising that the number of nonzero arrival probabilities increases with $\mathcal{U}$ since when the ground choice model has more diverse preference lists the fitted model will also need to account for a more diverse array of substitution patterns. There are, however, a few surprising takeaways from this table. First, given that the ground choice model contains 5000 customer classes and that 10000 customer arrivals is a substantial amount of sales data, one would expect the richer fitted models such as NP_4 and NP_5 to contain far more than 150 nonzero arrival probabilities. Further, it is interesting that the rate at which the number of nonzero arrival probabilities grows as a function of τ is fairly similar for NP_2, NP_3, and NP_4, even though the number of customer classes in NP_5 is a factor of $O(n^2)$ larger than NP_3. Across all three models, we essentially see this metric double as τ ranges from 1000 to 10000. This result is just one piece of evidence suggesting that the additional computation expense needed to fit increasingly complex nonparametric choice models does not provide a proportionate increase in fit.

The test cases with $\mathcal{U} = 4$ provide another setting in which fitting increasingly complex nonparametric choice models does not provide a significant benefit. We see that the assortments recommended by NP_4, NP_5 are only marginally more

profitable than the assortments recommended by NP_2 and less profitable than the assortments recommended by NP_3. When $\mathcal{U} \in \{6, 8\}$, the performance of NP_2 suffers in comparison to NP_4, NP_5, with the latter two fitted models providing assortments that can be over 6% more profitable than those recommended by NP_2. Nonetheless, when $\mathcal{U} = 6$, the performance of NP_3 is on par with that of NP_4 and NP_5 with regards to the revenues of the recommended assortments, which are never more than one percent more profitable. When $\mathcal{U} = 8$, NP_5 performs only marginally better than NP_4 on both metrics. Considering that the computation time required to fit NP_5 is at least an order of magnitude larger than the time required to fit NP_2, NP_3, and NP_4, it is clear that there is value in fitting nonparametric models with shorter preference lists. This is further validated by the fact that NP_3 or NP_4 perform better or comparable to MNL for most test cases under both metrics, and NP_2 produces more profitable assortments than MNL when $\mathcal{U} = 4$.

Finally, it is interesting to note that the MAE for all fitted models decreases as τ increases from 1000 to 10000. Moreover, the performance of the recommend assortments for each of the nonparametric fitted models, barring NP_1, also improves as τ increases. This seems to imply that more accurate fitted models for nonparametric choice models generally lead to more profitable assortments. However for the MNL fitted models, we see that increased levels of accuracy with respect to MAE have almost no affect on the profitability of the recommended assortments. This trend illustrates the fact that the revenue ordered assortments, which are well known to always be optimal for any MNL choice model, can be limiting. As a result, even if one can improve the fit of the MNL model with additional data, it is possible that the ground choice model could take a form for which revenue ordered assortments do not perform well.

$\mathcal{U}$	Model	τ 1000	5000	10000
4	NP_1	0.87	7.89	67.06
4	NP_2	1.44	17.54	38.04
4	NP_3	4.12	19.00	55.95
4	NP_4	10.17	219.07	521.31
4	NP_5	46.17	674.57	2243.01
6	NP_1	1.03	7.48	26.80
6	NP_2	1.53	18.16	40.71
6	NP_3	3.94	19.14	40.98
6	NP_4	11.58	227.66	465.21
6	NP_5	47.93	742.23	2574.16
8	NP_1	0.86	7.04	27.89
8	NP_2	1.41	17.94	40.28
8	NP_3	3.86	18.14	53.64
8	NP_4	9.71	206.08	402.46
8	NP_5	47.63	747.29	3434.05

(a) Average runtime (seconds) of the MLE procedures for each fitted model.

$\mathcal{U}$	Model	τ 1000	5000	10000
4	NP_1	10	10	10
4	NP_2	36.8	56.8	63.2
4	NP_3	54.5	90.4	109.0
4	NP_4	60.2	106.2	131.3
4	NP_5	64.7	115.2	137.9
6	NP_1	10	10	10
6	NP_2	35.5	54.3	61.8
6	NP_3	58.1	106.5	124.5
6	NP_4	67.1	126.3	150.8
6	NP_5	71.9	135.4	165.9
8	NP_1	10	10	10
8	NP_2	35.7	53.8	61.4
8	NP_3	58.3	105.1	129.6
8	NP_4	71.2	127.7	156.2
8	NP_5	80.1	146.8	177.9

(b) Average number of nonzero arrival probabilities in the fitted model.

Table 4.3: Other measures of the MLE procedure.

CHAPTER 5

VERTICALLY DIFFERENTIATED PRODUCTS

In this section, the choice model that we focus on is an extension of the classic linear utility model first seen in Mussa and Rosen [47], in which the random utility of each product is a linear function of the product's price and quality. The choice model we present is the sequential flips nonparametric choice model. In this setting, the set of potential preference lists is derived from an initial ordering of the products (including the no-purchase option) and additional preference lists are created by a series of sequential flips of adjacent products. We first show that the sequential flips nonparametric choice model is a more general form of the linear utility model by deriving the corresponding nonparametric preference lists from the orderings of the linear utility function; each time two of the utility functions intersect the ordering of the linear functions changes and we get a new preference list. As a result, the preference lists of consecutive customer types can be derived by flipping the rankings of two products. This new perspective on the linear utilities model allows us to develop a novel method to solve the associated assortment optimization and discrete pricing problems, further demonstrating the power and flexibility of the nonparametric choice model.

5.1 Model

We begin this section by formally introducing one of the most well-studied models for customers purchasing from a set of vertically differentiated products. In the linear utility model, arriving customers associate a linear random utility with each product and choose the product with highest positive utility. More formally, given a set $N = \{1, \ldots, n\}$ of vertically differentiated products we let the quality and

price of each product $i \in N$ be given by q_i and p_i, respectively. An arriving customer associates utility $U_i = \theta q_i - p_i$ with each product $i \in N$, where θ is a random variable with arbitrary distribution $F(\theta)$. This random variable reflects how each arriving customer values quality over price. The no-purchase option, also referred to as product 0, is assumed to have utility $U_0 = 0$. We assume customers are utility maximizing and will purchase the offered product with highest positive utility. In particular, if the retailer offers assortment $S \subseteq N$, then we can express the probability that an arriving customer purchases product $j \in S$ as $\Pr_j(S) = \Pr(U_j = \max_{i \in S \cup \{0\}} U_i)$.

In contrast, the sequential flips nonparametric model is a special case of the general nonparametric choice model. Slightly different from our other models, we assume that under this choice model a customer of class $g \in \mathcal{G}$ arrives with probability λ_g and is associated with a preference list given by an ordering σ_g on *all* products, including product 0. As before, an arriving customer will purchase the lowest indexed (most preferred) product offered, and we assume the no-purchase option is always available. In other words, any products ranked after product 0 are never considered.

Under the sequential flips nonparametric choice model, the set of customer classes can be indexed such that the preference list for customer class $g + 1$ can be derived from the preference list for customer class g by merely swapping the ordering of two adjacent products in customer class g's preference list. More formally, for customer class $g \in \mathcal{G}$, let products l_g and k_g satisfy $\sigma_g(l_g) = \sigma_g(k_g) - 1$ and assume that these are the two products that swap places to produce customer class $g + 1$'s preference list. Then, the preference list of customer class $g + 1$ can

be written as

$$\sigma_{g+1}(i) = \begin{cases} \sigma_g(i) & \text{if } i \notin \{l_g, k_g\} \\ \sigma_g(l_g) & \text{if } i = k_g \\ \sigma_g(k_g) & \text{if } i = l_g \end{cases}.$$

For the remainder of this paper, we assume that the customer classes are indexed as described above.

The second restriction we impose on the set of preference lists is that once one product swaps ahead of another product in a preference list, it can never swap back. In other words, for any pair of products $j, k \in N$, let $G(j,k) = \{g \in \mathcal{G} : \sigma_g(j) < \sigma_g(k)\}$ be the set of customer classes for which product j is preferred to product k. This restriction implies that $G(j,k)$ is a consecutive set of customer classes. Note that with this restriction in place, it is not hard to see that $m = O(n^2)$. Lastly, for ease of presentation, we assume that the no-purchase option never swaps above another product. This implies that if a customer of type g makes a purchase, then all customers of type $g' > g$ will also make a purchase. All results continue to hold without this last restriction.

5.1.1 Relationship to Linear Utility Model

Before diving into the corresponding optimization problems, we first prove that the sequential flips nonparametric choice model generalizes the linear utility choice model. Given any arbitrary linear utility choice model on n products with prices $p = \{p_1, \ldots, p_n\}$ and qualities $q = \{q_1, \ldots, q_n\}$, we define $\Theta(p,q) = \{\theta_1, \theta_1, \ldots, \theta_m, \theta_{m+1}\}$ to be the ordered values of $\theta > 0$ for which at least two products have the same utility, meaning there exists $i, j \in N$ such that $\theta q_i - p_i = \theta q_j - p_j$.

Further, we add the extreme points $\theta_1 = F^{-1}(0)$ and $\theta_{m+1} = F^{-1}(1)$ to represent the endpoints of the distribution of θ, possibly equal to $-\infty$ or ∞. Since the utilities are linear functions of θ, $m = O(n^2)$ since n lines have at most $O(n^2)$ intersection points. Consider the following set of intervals $\mathcal{I} = \{I_1, \ldots, I_m\}$, where $I_l = [\theta_l, \theta_{l+1}]$ for all $1 \leq l \leq m$. Note that, by construction, any customer that arrives with $\theta \in I_l$ will have the same ordering of product utilities. Let ρ_l give this ordering of the product utilities when $\theta \in I_l$, and let $\rho_l(i)$ give the index of product i in this ordering. Again, we use the convention that a lower index in this ordering implies a higher utility. See Figure 5.1 for a full visualization of this procedure. For a given linear utility choice model, the following theorem shows how to construct an equivalent sequential flips nonparametric choice model.

Proposition 5.1.1. *For arbitrary linear utility choice model with utility intersection points given by $\Theta(p,q) = \{\theta_1, \theta_2, \ldots, \theta_m, \theta_{m+1}\}$, we can construct an equivalent sequential flips nonparametric choice model with $\mathcal{G} = \{1, 2, \ldots, m\}$. The arrival probability of each customer class $l \in \mathcal{G}$ is $\lambda_l = P(\theta \in I_l)$ and the preference list for customer class l is created by setting $\sigma_l(i) = \rho_l(i)$ for all $i \in N$.*

Proof. Under this sequential flips nonparametric choice model, let $\hat{\Pr}_j(S)$ be the probability that product j is purchased when assortment $S \subseteq N$ is offered. Then,

$$\begin{aligned}
\hat{\Pr}_j(S) &= \sum_{l \in \mathcal{G}: \pi_l(S) = j} \lambda_l \\
&= \sum_{l=0}^{m} \mathbb{1}_{j = \text{argmin}_{i \in S \cup \{0\}} \sigma_l(i)} \Pr(\theta \in I_l) \\
&= \sum_{l=0}^{m} \mathbb{1}_{j = \text{argmax}_{i \in S \cup \{0\}} U_i} \Pr(\theta \in I_l) \\
&= \Pr(U_j = \max_{i \in S \cup \{0\}} U_i)
\end{aligned}$$

as desired. The first equality follows from the definition of λ_l, and the second

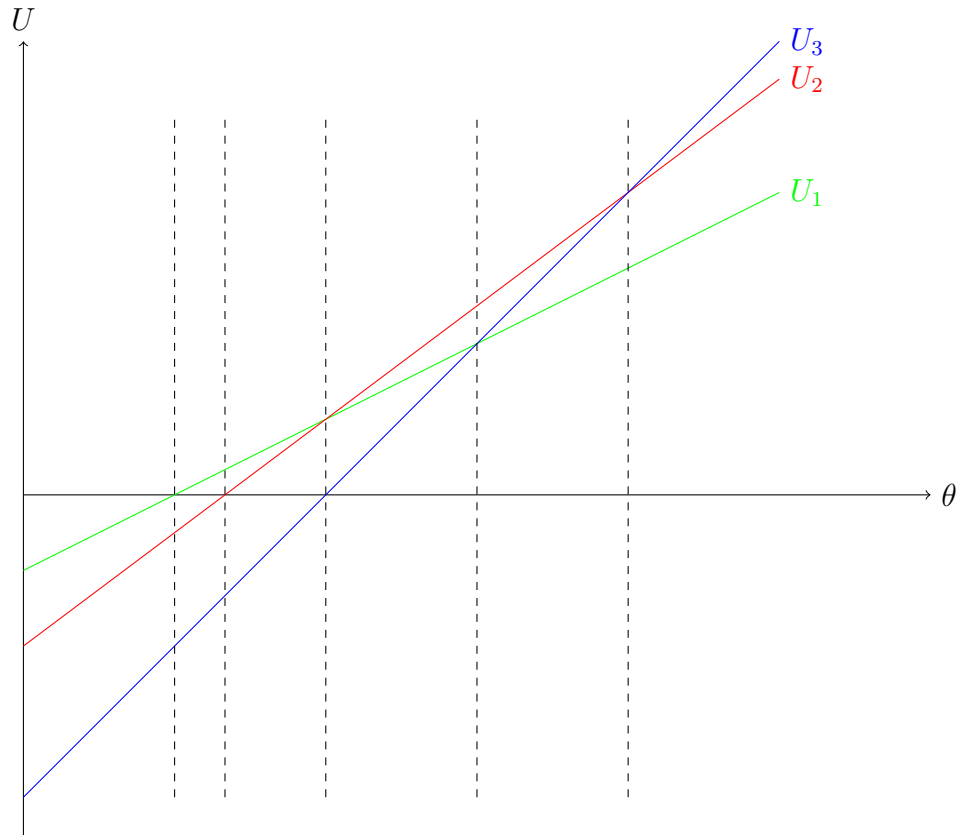


Figure 5.1: Visualization of the transformation from the linear utility model to the sequential flips nonparametric choice model for a 3 product example with 5 customer classes. Note that we have $\sigma_1 = [0, 1, 2, 3], \sigma_2 = [1, 0, 2, 3], \sigma_3 = [1, 2, 0, 3], \sigma_4 = [2, 1, 0, 3], \sigma_5 = [2, 1, 3, 0], \sigma_6 = [2, 3, 1, 0], \sigma_7 = [3, 2, 1, 0]$.

equality follows since the ordering of the products in customer class l's preference list is the same as the ordering of the utilities in I_l. □

There are several advantages to viewing customer choice on a set of vertically differentiated products through the lens of the nonparametric choice model. First, as Proposition 5.1.1 demonstrates, the sequential flips nonparametric choice model is a richer model for customer choice than the linear utility model. The sequential flips nonparametric choice model essentially allows us to generalize the utility function to relationships that are not linear. Specifically, the sequential flips non-

parametric model can capture any relationship in which the utility functions for every pair of products intersect at most once. This restriction is imposed by our assumptions that two products flip at most once. Further, for the sequential flips nonparametric choice model, we do not need to specify the distribution $F(\theta)$, which is likely difficult to tease out from historical sales data. Instead, we can input the qualities of each product and then use maximum likelihood estimation to estimate the distribution $F(\theta)$.

5.2 Assortment Optimization

We begin our analysis of the sequential flips nonparametric choice model by studying the assortment optimization problem. The authors of Pan and Honhon [49] show that the assortment problem under the linear utilities choice model can be recast as a shortest path problem. We boil down the insights used to create this shortest path problem into a simple dynamic program, which solves the assortment optimization problem and improves upon the runtime of Pan and Honhon [49] by a factor of n. In Section 5.3, we leverage this new dynamic programming approach to solve the discrete pricing and assortment problem under the linear utilities model.

Before we present the dynamic program, we derive an efficient way to compute the product that a particular customer class will purchase. For any assortment $S \subseteq N$, we first note that the product that customer class g purchases can be derived from the product purchased by customer class $g-1$ and knowing whether or not product k_g is offered. Recall that k_g is the product that flipped above product l_g to form customer class g. Therefore, if customer class $g-1$ purchases product $j \neq l_g$, then we know that customer class g must also purchase product j.

On the other hand, if customer class $g-1$ purchases product $j = l_g$, then customer class g will purchase product k_g if $k_g \in S$ and product j otherwise. In other words, for assortment $S \subseteq N$ we have that

$$\pi_g(S) = \begin{cases} \pi_{g-1}(S), & \text{if } k_g \notin S \text{ or } \pi_{g-1}(S) \neq l_g \\ k_g & \text{if } k_g \in S \text{ and } \pi_{g-1}(S) = l_g \end{cases}. \tag{5.1}$$

Building on this observation, it becomes fairly straightforward to develop a simple recursion for computing the revenue of any assortment. Let $R(S, g)$ be the revenue accrued from customer classes $g, g+1, \ldots, m$ from assortment $S \subseteq N$. For any assortment $S \subseteq N$ and customer class $g \in \mathcal{G}$, we can compute $R(S, g)$ through the following recursion:

$$R(S, g) = \begin{cases} \lambda_g r_{\pi_{g-1}(S)} + R(S, g+1), & \text{if } k_g \notin S \text{ or } \pi_{g-1}(S) \neq l_g \\ \lambda_g r_{k_g} + R(S, g+1) & \text{if } k_g \in S \text{ and } \pi_{g-1}(S) = l_g \end{cases}$$

with base case

$$R(S, m+1) = 0.$$

We can rewrite this recursion more concisely by aggregating consecutive customer classes that purchase the same product. To do so, we introduce the following two definitions for $g > 1$. Let

$$O(g) := \min\{g' > g : l_{g'} = k_g \text{ or } g = m+1\}$$
$$N(g) := \min\{g' > g : l_{g'} = l_g \text{ or } g = m+1\}.$$

There are a two important insights about $O(g)$ and $N(g)$. First, we have that $k_g = l_{O(g)}$ and $l_g = l_{N(g)}$, by construction. Further, note that if customer class g purchases product k_g, so will all customer classes indexed $g+1, \ldots, O(g)-1$. On the other hand, if customer class g purchases product l_g, we know that customer

classes indexed $g+1,\ldots,N(g)-1$ must also purchase l_g. For $g=1$, we develop a slightly modified version of $O(g)$ that is a function of the product purchased by the first customer class. Let N_1 be all products j such that $\sigma_1(j) \leq \sigma_j(0)$. This is the set of all products, including the no-purchase option, that the first customer class could have purchased. Then, for $j \in N_1$, we define

$$O(1,j) := \min\{g' > 1 : l_{g'} = j \text{ or } g' = m+1\}.$$

Building on this notation, we also find associated probabilities for these intervals in which customers purchase the same product. For each customer class g, we define

$$Q^o(g) := \sum_{q=g}^{O(g)-1} \lambda_q \quad \text{and} \quad Q^n(g) := \sum_{q=g}^{N(g)-1} \lambda_q,$$

and for each product $j \in N_1$, we define

$$Q^1(j) := \sum_{q=1}^{O(1,j)-1} \lambda_q.$$

These terms are useful for developing concise notation when computing expected revenues of assortments. Specifically, for any assortment $S \subseteq N$ and any customer class $g \in \mathcal{G}$ such that $\pi_{g-1}(S) = l_g$ we have that

$$R(S,g) = \begin{cases} r_{k_g} Q^o(g) + R(S, O(g)) & \text{if } k_g \in S \\ r_{l_g} Q^n(g) + R(S, N(g)), & \text{if } k_g \notin S \end{cases} \tag{5.2}$$

with base case

$$R(S, m+1) = 0.$$

This more concise recursion is critical in proving the correctness of the dynamic program that we develop next. The value function $V(g)$ in our dynamic program represents the maximum expected revenue accrued from customer classes $g, g+$

$1, \ldots, m$ when customer class $g-1$ purchases product l_g. The dynamic program is given below.

$$V(g) = \begin{cases} \max\{r_{k_g} Q^o(g) + V(O(g)), r_{l_g} Q^n(g) + V(N(g))\} & g > 1 \\ \max_{j \in N_1}\{r_j Q^1(j) + V(O(1,j))\} & g = 1 \end{cases} \tag{5.3}$$

with base case

$$V(m+1) = 0.$$

Note that $V(1)$ will be the maximum expected revenue from all customers and the maximization over $j \in N_1$ represents which product a customer of type $g = 1$ purchases (possibly equal to the no-purchase option). For $g > 1$, the first case of the maximum given in Equation 5.3 represents choosing to offer product k_g, implying that customer classes $g, g+1, \ldots, O(g)-1$ also purchase this product. The second case represents not offering product k_g and hence customer classes $g, g+1, \ldots, N(g)$ purchase product l_g. The following theorem shows that our value functions have the desired interpretation.

Theorem 5.2.1.

$$V(g) = \begin{cases} \max\limits_{\substack{S \subseteq N: \\ \pi_{g-1}(S) = l_g}} R(S,g) & \text{if } g > 1 \\ \max\limits_{S \subseteq N} R(S,1) & \text{if } g = 1 \end{cases}. \tag{5.4}$$

Proof. We show both cases in the theorem statement by induction. First, we consider the case in which $g > 1$. The result holds trivially for customer class $m+1$. Therefore, we assume inductively that the result holds for customer classes $g' > g$ and prove the result for g. Let S^* be the assortment that maximizes the right hand side of Equation 5.4 for customer class g.

Suppose that $k_g \in S^*$. Then, all customers from g to $O(g)-1$ purchase product k_g. Further, by our assumption that two products never flip twice, no customer

class $g' > g$ will purchase product l_g. In other words, for any subset $S \subseteq N$ such that $\pi_{O(g)-1}(S) = k_g$, $R(S, O(g)) = R(S \cup \{l_g\}, O(g))$. Therefore, given our assumption that $k_g \in S^*$, S^* also maximizes $R(S, O(g))$, and

$$\begin{aligned}\max_{\substack{S\subseteq N:\\ \pi_{g-1}(S)=l_g}} R(S,g) &= r_{k_g}Q^o(g) + \max_{\substack{S\subseteq N:\\ \pi_{O(g)-1}(S)=k_g}} R(S,O(g)) \\ &= r_{k_g}Q^o(g) + V(O(g)),\end{aligned}$$

where the first equality holds by our assumption of $k_g \in S^*$ and our observation above, and the second equality follows by the inductive hypothesis. Similarly, if $k_g \notin S^*$ then we have

$$\begin{aligned}\max_{\substack{S\subseteq N:\\ \pi_{g-1}(S)=l_g}} R(S,g) &= r_{l_g}Q^n(g) + \max_{\substack{S\subseteq N:\\ \pi_{N(g)-1}(S)=l_g}} R(S,N(g)) \\ &= r_{l_g}Q^n(g) + V(N(g)).\end{aligned}$$

By taking the maximum of the two cases above,

$$\max_{\substack{S\subseteq N:\\ \pi_{g-1}(S)=l_g}} R(S,g) = \max\{r_{k_g}Q^o(g) + V(O(g)), r_{k_g}Q^n(g) + V(N(g))\} = V(g),$$

where the second equality follows by definition of $V(g)$. For $g = 1$, a similar argument conditioning on the product purchased by the first customer class yields

$$\max_{S\subseteq N} R(S,1) = \max_{j\in N_1}\{r_jQ^1(j) + V(O(1,j))\} = V(1).$$

□

Theorem 5.2.1 immediately shows that $V(1) = \text{OPT}$, which proves the correctness of the dynamic program given in Equation 5.3. In the lemma below, we show that the value functions can be computed in $O(m)$ time.

Lemma 5.2.2. *The value functions of the presented dynamic program can be computed in $O(m)$ operations.*

Proof. To solve the dynamic program, we will first calculate all $N(g)$, $O(g)$, $Q^o(g)$, $Q^n(g)$, and $Q^1(j)$ values in pre-processing. To do so, for each product i, we will keep a stack of tuples (g, F_g, I) in which g will be a customer class, $F_g = \sum_{i=1}^{g-1} \lambda_g$, and I will either be O or N. Initially, all stacks will be empty and we will set $F = 0$. For $g = 1, 2, \ldots m$, we first empty the stack for l_g.

- If the stack for l_g is already empty, we set $Q^1(l_g) = F$ and $O(1, l_g) = g$.
- Otherwise, for each $(g', F_{g'}, O)$ on the stack for l_g, we set

$$Q^o(g') = F - F_{g'}$$

and for each $(g', F_{g'}, N)$ on the stack for l_g, we set

$$Q^n(g') = F - F_{g'}.$$

We then add (g, F, O) to the stack for product k_g and add (g, F, N) to the stack for product l_g. Lastly, we set $F = F + \lambda_g$ before moving to the next customer class.

For each customer class g, (g, F_g, O) is added to the stack when processing g and it is removed the first time $l_{g'} = k_g$, and (g, F_g, N) is added to the stack when processing g and removed the first time $l_{g'} = l_g$. Therefore, all $O(g)$, $Q^o(g)$, $N(g)$, and $Q^n(g)$ values are calculated correctly. Similarly, $Q^1(j)$ is set the first time $l_g = j$ so $O(1, j)$ and $Q^1(j)$ are set correctly. Further, the pre-processing iterates through all customer classes and adds/removes each element to/from the stack exactly once. Therefore, the overall running time is $O(m)$.

After pre-processing, calculating all $V(g)$ values simply involves iterating through all $O(m)$ states and taking the maximum over two possible values in

constant time. This gives an overall running time of $O(m)$ for the assortment optimization algorithm. □

Our final runtime of $O(m) = O(n^2)$ improves upon the best previous runtime of $O(n^3)$ for the unconstrained assortment problem under the linear utilities model achieved by Pan and Honhon [49]. Considering that it takes $O(m)$ operations to compute the revenue of any assortment, it is likely that our approach achieves the best possible runtime.

5.2.1 Additional Costs

The dynamic program above extends naturally to the setting in which products have additional fixed costs. As in Chapter 3, we consider a setup cost incurred when offering a product and a substitution penalty incurred when a customer is forced to substitute to less desirable products. To model setup costs we introduce a constant fixed cost of k_i for offering product i, which could represent a setup or stocking cost. To model the substitution penalty we introduce a function $f(l)$ that represents the penalty incurred when a customer purchases their l^{th} most preferred product. Specifically, if a customer type g purchases product i and $l = \sigma_g(i)$ then the retailer incurs a penalty of $f(l)$. Below we present an extension of our dynamic program that includes these costs.

Similar to the $Q^o(g)$, $Q^n(g)$, and $Q^1(g)$ values above, we define penalty functions

$$P^o(g) := \sum_{q=g}^{O(g)-1} \lambda_q f(\sigma_g(k_g)) \quad \text{and } P^n(g) := \sum_{q=g}^{N(g)-1} \lambda_q f(\sigma_g(l_g)),$$

and for each product $j \in N_1$, we define

$$P^1(j \neq 0) := \sum_{q=1}^{O(1,j)-1} \lambda_q f(\sigma_g(j)) \quad \text{and} \quad P^1(0) = 0.$$

These terms calculate the sum of penalty functions. We can now write the modified dynamic program values:

$$V(g > 1) = \max\{r_{k_g} Q^o(g) - k_{k_g} - P^o(g) + V(O(g)), r_{l_g} Q^n(g) - P^n(g) + V(N(g))\}$$

$$V(1) = \max_{j \in N_1}\{r_j Q^1(j) - k_j - P^1(j) + V(O(1,j))\}$$

with base case

$$V(m+1) = 0.$$

The proof of correctness of the dynamic program is similar to the proof in Lemma 5.2.1. The only difference to the equations in the dynamic program given in Equation 5.3 is that we have added the fixed cost k_{k_g} if we decide to offer k_g and we subtract the penalty functions. Each $P^o(g)$, $P^n(g)$, and $P^1(j)$ value can be computed in $O(n+m)$ time and there are $O(m)$ values to compute. As a consequence, the running time of this modified algorithm is $O(m^2 + m)$.

5.2.2 Cardinality Constraints

Realistically, retailers are under many constraints and are not able to offer an arbitrary set of products to their customers. In the simplest case, a single shelf space constraint, each item consumes a single unit of capacity and the retailer has $C \leq n$ units of capacity on her shelves. This constraint is akin to a limit on the total number of products that the retailer can offer. Recall the cardinality constrained assortment optimization problem:

$$\max_{S:|S| \leq C} \text{Rev}(S). \tag{5.5}$$

We extend our dynamic programming approach to solve the capacitated version of the problem by adding in a state c that is the remaining number of products that we have left to offer. Our value function $V(g, c)$ will be the maximum expected revenue accrued from customer classed $g, g+1, \ldots, m$ when customer class $g-1$ purchases product l_g and we can offer at most c products. We can now write the modified dynamic program:

$$V(g,c) = \begin{cases} \max\{r_{k_g} Q^o(g) + V(O(g), c-1), r_{l_g} Q^n(g) + V(N(g), c)\} & g > 1 \\ \max_{j \in N_1}\{r_j Q^1(j) + V(O(1,j), c - 1_{j \neq 0})\} & g = 1 \end{cases} \tag{5.6}$$

with base cases

$$V(g, 0) = 0 \quad \text{and} \quad V(m+1, c) = 0.$$

The proof of correctness of the dynamic program is similar to the proof in Lemma 5.2.1. The only difference to the equations in DP 5.3 is that we have added state space keeping track of how many remaining products we can offer. If we offer k_g, we recurse with $c-1$ remaining products. Adding the extra state space, we now need to calculate $V(g, c)$ for every customer class g and value of c. After pre-processing the Q values in $O(m)$ time, each $V(g, c)$ takes constant time to compute yielding an overall runtime of $O(C \cdot m) = O(n \cdot m)$.

5.3 Joint Assortment and Pricing Optimization

We now extend our dynamic programming approach to solve the discrete pricing problem under the linear utility model. In this setting, the retailer also has control over the price p_i for each product i can set p_i to any value within a set of discrete prices $\Delta = \{\delta_1, \delta_2, \ldots, \delta_r\}$. For example, these discrete prices might represent five

dollar increments. Again, once a retailer has set prices a customer will purchase the product with highest linear utility. If product j is priced at δ_k, then the customer gains random utility $\theta q_j - \delta_k$ from that product. Pan and Honhon [49] study the *continuous* pricing version of this problem under the linear utilities choice model and provide an optimal algorithm for pricing that relies heavily on the distribution $F(\theta)$ and the flexibility of arbitrary prices. The algorithm does not work when prices must come from a discrete set of points. In contrast, the approach we present is novel and our procedure makes no assumption about the distribution $F(\theta)$.

To solve this discrete pricing problem, we will use a common technique (see [13] and [14]) to cast the problem as an assortment problem with $n \cdot r$ dummy products that represent each of the products priced at each of the different price levels. Specifically, for each product $i \in N$ and price point $\delta_j \in D$, we construct a dummy product represented by the tuple (i, δ_j) with associated revenue δ_j. Let $N' = \{(1, \delta_1), \ldots, (1, \delta_r), \ldots, (n, \delta_1), \ldots, (n, \delta_r)\}$ be the set of all dummy products.

Preference lists in the associated sequential flips model with discrete pricing will also consist of these dummy products so that the choice process works exactly as before. Namely, each arriving customer will purchase the highest ranking tuple (i, δ_j) in her preference list such that product i is offered at price δ_j. Similar to Section 5.2, for any pair of products $(i, \delta_j), (k, \delta_l) \in N$, we again impose that these products can flip at most once. Further, we add the new constraint that $\sigma_g((i, \delta_j)) < \sigma_g((i, \delta_{j+1}))$ for all $g \in \mathcal{G}$ and $j = 1, \ldots, r-1$. This restriction is natural and simply ensures that every customer always prefers to receive the same product at a lower price.

We let the set $\mathcal{S}_p = \{S \subset N' : |S \cap \{(i, \delta_1), \ldots, (i, \delta_r)\}| \leq 1 \; \forall \; i \in S\}$ be the set

of feasible pricing policies in which each product is associated with at most one price. Then, the expected revenue for an assortment $S \in \mathcal{S}_p$ under the sequential flips model with discrete pricing can be written as $\text{Rev}(S) = \sum_{(i,\delta_j)\in S} \Pr_j(S)\delta_j$ and so the joint assortment and pricing problem can be expressed as

$$\text{OPT}_p = \max_{S\in\mathcal{S}_p} \text{Rev}(S). \tag{5.7}$$

Here, we have ignored fixed costs for offering each product but the result extends to this setting as well.

To transform the linear utilities model with discrete pricing to this new sequential flips model we use a similar approach to that used to prove Proposition 5.1.1. Given any arbitrary linear utilities choice model on n products with prices $\Delta = \{\delta_1, \delta_2, \ldots, \delta_r\}$ and qualities $q = \{q_1, \ldots, q_n\}$, we define $\Theta(\Delta, q) = \{\theta_1, \theta_2, \ldots, \theta_m, \theta_{m+1}\}$ to be the ordered values of $\theta > 0$ for which at least two products have the same utility, meaning there exist two products $i, j \in N$ and prices $p_i, p_j \in \Delta$ such that $\theta q_i - p_i = \theta q_j - p_j$. Again, we add $\theta_1 = F^{-1}(0)$ and $\theta_{m+1} = F^{-1}(1)$. Since the utilities are linear functions of θ, we know that $m = O(n^2r^2)$ since nr lines intersect at most $O(n^2r^2)$ times. We construct the same set of intervals $\mathcal{I} = \{I_1, I_2, \ldots, I_m\}$, where $I_l = [\theta_l, \theta_{l+1}]$ for all $1 \leq l \leq m$. Again, any customer arriving with $\theta \in I_l$ will have the same ordering of the product utilities for all products priced at all possible price points. Let ρ_l give the ordering of the products and price tuples by utility when $\theta \in I_l$, and let $\rho_l((i, \delta_j))$ give the index of tuple (i, δ_j) in this ordering. Again, we use the convention that lower indices in this ordering have higher utilities. The following proposition shows that we can use these rankings to construct an equivalent sequential flips nonparametric choice model with discrete pricing.

Proposition 5.3.1. *For an arbitrary linear utilities choice model with utility*

intersection points given by $\Theta(\Delta, q) = \{\theta_1, \ldots, \theta_m, \theta_{m+1}\}$ *and prices given by* $\{\delta_1, \delta_2, \ldots, \delta_r\}$*, we can construct an equivalent sequential flips nonparametric choice model with discrete pricing in which the set of customer classes is given by* $\mathcal{G} = \{1, \ldots, m\}$.

Proof. The proof mirrors the one for the pure assortment setting. In particular, the arrival probability of each customer class $l \in \mathcal{G}$ is $\lambda_l = \Pr(\theta \in I_l)$ and the preference list for customer class l is created by setting set $\sigma_l((i, \delta_j)) = \rho_l(i, \delta_j)$ for all $(i, \delta_j) \in N'$. □

Given this reduction, we solve the discrete joint pricing and assortment problem through the dynamic programming approach in Section 5.2. As shown previously, this requires $m = O(n^2 r^2)$ time. However, there is an additional step in proving its validity in this case in that we must show that the dynamic program chooses an assortment in $\mathcal{S}_p$. In other words, we must show that the dynamic program prices each product at most once.

Proposition 5.3.2. $V(1)$ *calculates the optimal pricing policy for the discrete joint pricing and assortment problem under the sequential flips model.*

Proof. Given the correctness of the dynamic program for the assortment optimization problem, we just need to show that if an optimal assortment contains the tuple (i, δ_j) then it will never contain the tuple (i, δ_k) with $k > j$. This follows from the fact that $\sigma_g(i, \delta_j) \leq \sigma_g(i, \delta_k)$ for all $g \in \mathcal{G}$. Therefore, if both tuples are offered then (i, δ_k) is never purchased and can be removed from the assortment without affecting the overall revenue. Thus, an optimal assortment gives a feasible pricing policy. □

Lastly, we prove one additional structural result for the optimal pricing structure when choice is governed by the linear utilities choice model. The following proposition states that the optimal set of discrete prices are quality consistent. This result is also shown by Pan and Honhon [49] for the continuous joint pricing and assortment problem.

Proposition 5.3.3. *For the discrete pricing and assortment problem under the linear utilities choice model, there exists an optimal pricing policy $S \in \mathcal{S}_p$ such that if $(i, \delta_{l_1}), (j, \delta_{l_2}) \in S$ then $\delta_{l_1} < \delta_{l_2} < \infty$, then we also have $q_i \leq q_j$.*

Proof. Consider an optimal pricing policy $S \in \mathcal{S}_p$. Next consider any pair $(i, \delta_{l_1}), (j, \delta_{l_2}) \in S$ such that $\delta_{l_1} < \delta_{l_2}$. Suppose by way of contradiction that $q_i > q_j$. Then for all $\theta \geq 0$, we have $\theta q_i - \delta_{l_1} \geq \theta q_j - \delta_{l_2}$ and thus we can remove product j from the assortment without affecting any purchase probabilities. $\square$

CHAPTER 6

GENERAL SPACE CONSTRAINED ASSORTMENT OPTIMIZATION

In this chapter, we present a general two-stage approximation algorithm for the space constrained assortment optimization problem. The algorithm will apply to two of the models presented in the previous chapters as well as other random utility models.

6.1 Model and Hardness

Recall that in the space constrained version of the assortment optimization problem, each product i is also associated with a size $c_i \geq 0$, and the retailer wants to offer an assortment $S \subseteq N$ to maximize expected revenue with the added constraint that $\sum_{i \in S} c_i \leq C$, where C is a fixed constant. We define $C(S) := \sum_{i \in S} c_i$ and let OPT_c be the optimal expected revenue under this variant. The approach we present actually works generally for a large class of random utility models since we rely primarily on the following fact.

Fact 6.1.1. *Under any random utility model, for any assortments S_1 and S_2 such that $S_1 \subseteq S_2$ and $j \in S_1 \cap S_2$, $\Pr_j(S_1) \geq \Pr_j(S_2)$.*

Specifically, for any choice model under which Fact 6.1.1 holds, we show that if there exists a δ-approximation algorithm for the fixed cost assortment optimization problem, then there exists a $[3(1+\epsilon)\delta]$-approximation algorithm for the corresponding space constrained assortment optimization problem. Recall that in the fixed cost assortment optimization problem each product i is associated with

a fixed cost $k_i \geq 0$ that the retailer must pay if that product is offered, and the retailer wants to find the assortment S that maximizes $\max_{S \subseteq N}[\text{Rev}(S) - \sum_{i \in S} k_i]$.

First, we show that the space constrained assortment optimization problem is NP-hard under all three models presented in the previous chapters.

Theorem 6.1.2. *The space constrained assortment optimization problem is NP-hard under the nonparametric tree choice model, the k-product nonparametric choice model, and the sequential flips nonparametric choice model.*

Proof. For all three models we use a reduction from the knapsack problem. Given an instance of the knapsack problem with n items, let $v_1, \ldots, v_n$ be the associated item values, let $w_1, \ldots, w_n$ be the associated item sizes, and let W be the size of the knapsack. The goal of the knapsack problem is to find a subset of items of maximum total value with the constraint that the total size of the subset is at most W. We assume that the items are indexed such that $v_1 \leq v_2 \leq \ldots \leq v_n$.

For the nonparametric tree choice model and k-product nonparametric choice model, we create products $N = \{1, 2, \ldots, n\}$ corresponding to each item and let $r_i = v_i$ and $c_i = w_i$ for all $i \in N$. Further, we create a preference list $\sigma_g = [i]$ for each product $i \in N$ with unnormalized probability $\lambda_g = 1$. Last, we set $C = W$. For the nonparametric tree choice model, any tree on the products is consistent with these preference lists. For the k-product nonparametric choice model, these preference lists are all of length $\leq k$. In both cases, it is clear that the revenue of an assortment is exactly equal to the value of the corresponding items and the space equal to the weight, and vice versa. Therefore, solving the space constrained assortment optimization problem is equivalent to solving the original knapsack problem.

For the sequential flips nonparametric choice model, the reduction requires a bit more creativity. First, we create $2n-1$ products indexed by the set $N = \{1, 2, \ldots, 2n-1\}$ and set $M > \max_i v_i$. For odd $i \in N$, we set $r_i = v_{(i+1)/2} + M \cdot (i-1)/2$ and set $c_i = w_{(i+1)/2}$. For even $i \in N$, we set $r_i = M \cdot i/2$ and set $c_i = 0$. We start with the preference list $[2n-1, 2n-2, \ldots, 1, 0]$. Then, we flip product $2n-1$ down until it flips with product 0. This creates the preference list $[2n-2, 2n-3, \ldots, 0, 2n-1]$. We continue to iteratively flip the current first choice product down until it flips with product 0 to create $2n-1$ customer classes with preference lists of the form $[i, i-1, \ldots, 1, 0, \ldots]$ for $i = 1, \ldots, 2n-1$ each arriving with unnormalized probability $\lambda_g = 1$. All other generated preference lists have arrival probability $\lambda_g = 0$. Last, we set $C = W$.

We first show that it is always optimal to offer all even indexed products. To see this, note that an even indexed product i only blocks products $1, \ldots, i-1$, which all have lower revenues than product i, by construction. Further, product i does not consume any space. Conditioned on the fact that even indexed products are always offered, we show that adding an odd indexed product i generates an additional revenue of $v_{(i+1)/2}$. Specifically, for any assortment S containing all even indexed products and $i \notin S$, we have

$$\text{Rev}(S \cup \{i\}) - \text{Rev}(S) = v_{(i+1)/2} + M \cdot (i-1)/2 - M \cdot (i-1)/2 = v_{(i+1)/2}.$$

This follows from the fact that the only customer class to switch products will be the customer class with product i as their first choice product. Further, adding product i to the assortment takes up additional space $c_i = w_{(i+1)/2}$. Thus, the problem of determining which additional products to offer (beyond all even indexed products) is exactly the original knapsack problem. □

6.2 Two-Stage Algorithm

Motivated by our hardness result, we proceed to develop an approximation scheme for the space constrained assortment problem. We start by assuming that we can find two assortments S_1 and S_2 along with a weight $0 \leq \alpha \leq 1$ such that

$$\alpha \text{Rev}(S_1) + (1-\alpha)\text{Rev}(S_2) \geq \beta \cdot \text{OPT}_c$$

and

$$\alpha C(S_1) + (1-\alpha)C(S_2) \leq C.$$

In other words, if the retailer offers S_1 with probability α and S_2 with probability $1-\alpha$, then she obtains at least β times the optimal expected revenue in expectation and also satisfies the space constraint in expectation. We start by showing how to derive a new assortment from S_1 and S_2 with expected revenue at least $\beta \cdot \text{OPT}_c/3$ and size at most C. Next, we show two methods to find S_1, S_2, and α. Our first approach considers the Lagrangian relaxation of the space constrained problem, which turns out to be exactly the fixed cost version of the assortment problem. The second approach relies on a linear programming formulation for the unconstrained assortment problem.

Suppose that we have found S_1, S_2, and α as described above. We assume without loss of generality that $C(S_1) \leq C(S_2)$ and hence the assortment S_1 is feasible for the space constrained problem. Thus, we have found one initial feasible solution with expected revenue $\text{Rev}(S_1) \geq \alpha \text{Rev}(S_1)$. Next, we show how to find an assortment $\hat{S}$ such that $\text{Rev}(\hat{S}) \geq (1-\alpha)\text{Rev}(S_2)/2$. Consider the following linear program, which can be viewed as a fractional knapsack problem.

$$\begin{aligned} \text{maximize} \quad & \sum_{j \in S_2} r_j \Pr_j(S_2) x_j \\ \text{subject to} \quad & \sum_{j \in S_2} c_j x_j \leq C \\ & 0 \leq x_j \leq 1 \qquad \forall j \in S_2 \end{aligned}$$

The following proposition shows that we can construct a feasible solution to this linear program with objective function value $\geq (1-\alpha)\text{Rev}(S_2)$.

Lemma 6.2.1. *The solution $\tilde{x}_j = 1 - \alpha$ for all $j \in S_2$ is a feasible solution to the knapsack linear program and has objective function value equal to $(1-\alpha)\text{Rev}(S_2)$.*

Proof. First we show that the proposed solution is feasible by considering its space consumption.

$$\begin{aligned} \sum_{j \in S_2} c_j \tilde{x}_j &= (1-\alpha) \sum_{j \in S_2} c_j \\ &= (1-\alpha) C(S_2) \leq C. \end{aligned}$$

The inequality follows from the properties of S_1 and S_2. Thus, $\tilde{x}_j$ is a feasible solution to the knapsack linear program. Further, its objective function value is

$$\begin{aligned} \sum_{j \in S_2} r_j \Pr_j(S_2) \tilde{x}_j &= (1-\alpha) \sum_{j \in S_2} r_j \Pr_j(S_2) \\ &= (1-\alpha)\text{Rev}(S_2). \end{aligned}$$

□

Lemma 6.2.1 shows that the optimal objective function value of the knapsack linear program is at least $(1-\alpha)\text{Rev}(S_2)$. On the other hand, it is well known that the optimal solution to the linear programming relaxation of any knapsack

problem has at most one fractional variable value. Let x^* be an optimal solution to the linear programming relaxation, and let $\hat{S}_1 = \{j \in S_2 : x_j^* = 1\}$ and $\hat{S}_2 = \{j \in S_2 : 0 < x_j^* < 1\}$. Note that $|\hat{S}_2| \leq 1$. The following lemma bounds the revenue of the best of these two solutions.

Lemma 6.2.2. *Let* $\hat{S} = \text{argmax}_{S \in \{\hat{S}_1, \hat{S}_2\}} \text{Rev}(S)$. *Then,* $C(\hat{S}) \leq C$ *and*

$$\text{Rev}(\hat{S}) \geq \frac{1}{2}(1-\alpha)\text{Rev}(S_2).$$

Proof. By the feasibility of x^*, $C(\hat{S}_1) \leq C$. Further, without loss of generality, every product has size at most C. Therefore, $C(\hat{S}_2) \leq C$ as well, and $C(\hat{S}) \leq C$. Next we consider the revenue of assortment $\hat{S}$,

$$\begin{aligned}
\text{Rev}(\hat{S}) &= \max\{\text{Rev}(\hat{S}_1), \text{Rev}(\hat{S}_2)\} \\
&\geq \max\left[\sum_{j \in \hat{S}_1} r_j \text{Pr}_j(S_2), \sum_{j \in \hat{S}_2} r_j \text{Pr}_j(S_2)\right] \\
&\geq \frac{1}{2}\left[\sum_{j \in S_2} r_j \text{Pr}_j(S_2) x_j^*\right] \\
&\geq \frac{1}{2}(1-\alpha)\text{Rev}(S_2).
\end{aligned}$$

The first inequality comes from the fact that $\hat{S}_1, \hat{S}_2 \subseteq S_2$ and hence we can apply Fact 6.1.1. The second inequality results from the fact that

$$\sum_{j \in \hat{S}_1} r_j \text{Pr}_j(S_2) + \sum_{j \in \hat{S}_2} r_j \text{Pr}_j(S_2) \geq \sum_{j \in S_2} r_j \text{Pr}_j(S_2) x_j^*.$$

Finally, the last inequality follows from Lemma 6.2.1. □

At this point, we consider either offering the assortment S_1 or $\hat{S}$. The following theorem shows that returning the best of these assortments produces an assortment S^* with revenue $\text{Rev}(S^*) \geq \beta \cdot \text{OPT}_c/3$.

Theorem 6.2.3. *Let* $S^* = \text{argmax}_{S \in \{S_1, \hat{S}\}} \text{Rev}(S)$. *Then,*

$$\text{Rev}(S^*) \geq \beta \cdot \text{OPT}_c/3.$$

Proof. As constructed, $\text{Rev}(S^*) = \max\{\text{Rev}(S_1), \text{Rev}(\hat{S})\} \geq \max\{\alpha \text{Rev}(S_1), (1-\alpha)\text{Rev}(S_2)/2\}$. Recall that $\alpha \text{Rev}(S_1) + (1-\alpha)\text{Rev}(S_2) \geq \beta \cdot \text{OPT}_c$. If $\alpha \cdot \text{Rev}(S_1) \geq \frac{1}{3}\beta \cdot \text{OPT}_c$, then the theorem holds. Otherwise, $(1-\alpha) \cdot \text{Rev}(S_2) \geq \frac{2}{3}\beta \cdot \text{OPT}_c$ and again the theorem holds. □

6.2.1 Finding a Convex Combination of Assortments

Finally, we show two methods to find S_1, S_2, and α such that $\alpha \text{Rev}(S_1) + (1-\alpha)\text{Rev}(S_2) \geq \beta \cdot \text{OPT}_c$ for some $\beta \geq 0$ and $\alpha C(S_1) + (1-\alpha)C(S_2) \leq C$. First, we assume that for a given customer choice model we have access to a δ-approximation algorithm for the fixed cost assortment optimization problem. We consider the Lagrangian relaxation of the space constrained assortment optimization problem in which we associate a Lagrangian multiplier $\lambda \geq 0$ with the space constraint. This problem is formulated below:

$$\text{OPT}_\lambda = \max_{S \subseteq N} \text{Rev}(S) + \lambda(C - C(S)) = \max_{S \subseteq N} \text{Rev}(S) - \lambda C(S) + \lambda C. \tag{6.1}$$

First, note that for fixed $\lambda \geq 0$, this problem is equivalent to the fixed cost assortment optimization problem with $k_i = \lambda c_i$ and hence we can find an assortment S_λ in polynomial time such that

$$\text{Rev}(S_\lambda) - \lambda C(S_\lambda) + \lambda C \geq (1/\delta) \cdot \text{OPT}_\lambda.$$

Note that for any $\lambda \geq 0$, $\text{OPT}_\lambda \geq \text{OPT}_c$ since the optimal solution to the space constrained assortment optimization problem is also a feasible solution to the fixed

cost assortment problem with at least as high revenue. Suppose that we can find $\lambda^* \geq 0$ such that $C(S_{\lambda^*}) = C$. Then,

$$\text{Rev}(S_{\lambda^*}) = \text{Rev}(S_{\lambda^*}) - \lambda^* C(S_{\lambda^*}) + \lambda^* C \geq \frac{1}{\delta} \cdot \text{OPT}_{\lambda^*} \geq \frac{1}{\delta} \cdot \text{OPT}_c.$$

This implies that we are within a factor of $1/\delta$ of the optimal revenue for the space constrained assortment problem. We show that we either we can find λ^* or we can find S_{λ_1} and S_{λ_2} along with a carefully chosen α such that $\alpha\text{Rev}(S_1) + (1 - \alpha)R(S_2) \geq \frac{1}{\delta(1+\epsilon)} \cdot \text{OPT}_c$ for any $\epsilon \geq 0$. Hence, in the worst case, we produce an assortment with an approximation guarantee of $3(1 + \epsilon)\delta$.

We find S_{λ_1} and S_{λ_2} through bisection search on the interval $[0, r_{\max}/c_{\min}]$, where $r_{\max} = \max_{i \in N} r_i$ and $c_{\min} = \min_{i \in N} c_i$. Initially, we set $\lambda_1 = 0$ and $\lambda_2 = r_{\max}/c_{\min}$. Note that for $\lambda = \lambda_1$ the problem reduces to the unconstrained assortment problem and $C(S_{\lambda_1}) > C$. On the other hand, for $\lambda = \lambda_2$ the penalty for offering any subset is always at least the revenue generated by that subset and $S_{\lambda_2} = \emptyset$. Consequently, $C(S_{\lambda_2}) < C < C(S_{\lambda_1})$. Given λ_1 and λ_2, in each iteration the algorithm tests the midpoint $\lambda' = \frac{1}{2}(\lambda_1 + \lambda_2)$. If $C(S_{\lambda'}) = C$, then we return $S_{\lambda'}$. Otherwise, if $C(S_{\lambda'}) > C$, then we set $\lambda_1 = \lambda'$, and if $C(S_{\lambda'}) < C$, then we set $\lambda_2 = \lambda'$. We repeat this process until either we find an assortment with a space consumption of exactly C or until $\lambda_2 - \lambda_1 \leq \epsilon \cdot \text{Rev}_{\min}/\mathcal{C}$, where $\text{Rev}_{\min} = \min_{i \in N} r_i \cdot \min_{g \in \mathcal{G}} \lambda_g$ is a lower bound on the revenue of any offered assortment and $\mathcal{C} = \sum_{i \in N} c_i$. Without loss of generality, $\text{Rev}_{\min} > 0$ since we can remove any product i with revenue $r_i = 0$ and any customer class with probability $\lambda_g = 0$ from consideration. We note that this implies that there are at most $O(\log \frac{\mathcal{C} \cdot r_{\max}}{\epsilon \text{Rev}_{\min} \cdot c_{\min}})$ calls to the fixed cost assortment problem, which is polynomial in the input size.

Given $S_1 = S_{\lambda_1}$ and $S_2 = S_{\lambda_2}$ as constructed above, we set

$$\alpha = \frac{C - C(S_{\lambda_2})}{C(S_{\lambda_1}) - C(S_{\lambda_2})}$$

which implies

$$1 - \alpha = \frac{C(S_{\lambda_1}) - C}{C(S_{\lambda_1}) - C(S_{\lambda_2})}.$$

By construction of S_1 and S_2, $0 \leq \alpha \leq 1$. We now prove that S_1, S_2, and α satisfy the properties we are looking for in order to find S^*.

Lemma 6.2.4. *Given assortments S_1 and S_2 and weight α, we have that $\alpha\text{Rev}(S_1) + (1-\alpha)\text{Rev}(S_2) \geq \frac{1}{\delta(1+\epsilon)} \cdot \text{OPT}_c$ and $\alpha C(S_1) + (1-\alpha)C(S_2) \leq C$.*

Proof. First, we note that

$$\alpha C(S_1) + (1-\alpha)C(S_2) = \frac{C - C(S_2)}{C(S_1) - C(S_2)} C(S_1) + \frac{C(S_1) - C}{C(S_1) - C(S_2)} C(S_2) = C.$$

Second, we analyze the revenue of the convex combination. For any $\lambda \geq 0$, we note that $\text{OPT}_\lambda \geq \text{OPT}_c$ since any feasible assortment only has higher revenue in this fixed cost problem. Therefore,

$$\begin{aligned}
\frac{1}{\delta}\text{OPT}_c &\leq \frac{1}{\delta}\left[\alpha\text{OPT}_{\lambda_1} + (1-\alpha)\text{OPT}_{\lambda_2}\right] \\
&\leq \alpha\text{Rev}(S_1) + (1-\alpha)\text{Rev}(S_2) + \alpha\lambda_1[C - C(S_1)] + (1-\alpha)\lambda_2[C - C(S_2)] \\
&= \alpha\text{Rev}(S_1) + (1-\alpha)\text{Rev}(S_2) + \lambda_2\left(\alpha[C - C(S_1)] + (1-\alpha)[C - C(S_2)]\right) \\
&\quad - (\lambda_2 - \lambda_1)\alpha(C - C(S_1)) \\
&= \alpha\text{Rev}(S_1) + (1-\alpha)\text{Rev}(S_2) - (\lambda_2 - \lambda_1)\alpha(C - C(S_1)) \\
&\leq \alpha\text{Rev}(S_1) + (1-\alpha)\text{Rev}(S_2) + (\lambda_2 - \lambda_1)\mathcal{C} \\
&\leq \alpha\text{Rev}(S_1) + (1-\alpha)\text{Rev}(S_2) + \epsilon\text{Rev}_{\min} \\
&\leq \alpha\text{Rev}(S_1) + (1-\alpha)\text{Rev}(S_2) + \epsilon\min(\text{Rev}(S_1), \text{Rev}(S_2)),
\end{aligned}$$

where the second to last inequality follows from our stopping criterion on the difference between λ_2 and λ_1. □

Theorem 6.2.5. *There exists a $3(1+\varepsilon)$-approximation algorithm for the space constrained assortment optimization problem under the nonparametric tree choice model and under the sequential flips nonparametric choice model.*

Proof. In Section 3.3.1, we present a dynamic program for the fixed cost assortment optimization problem under the nonparametric tree choice model, and in Section 5.2.1 we do the same for the sequential flips nonparametric choice model. Running the Lagrangian relaxation method above and applying Theorem 6.2.3 gives the result. □

For certain problems, there exists a simpler method to find S_1 and S_2 that avoids doing bisection search. Suppose that one can find a linear programming relaxation of the unconstrained assortment optimization problem such that all basic feasible solutions are integral and have objective function value equal to their associated revenue. Then, by adding in the space consumption constraint to the linear program, an optimal solution to the new linear program lies on an edge of the previous polytope and is a convex combination of two integral assortments S_1 and S_2. Further, we can find this convex combination in polynomial time. Setting α to be the convex combination coefficient for S_1, it follows from the linearity of the solution that $\alpha C(S_1)+(1-\alpha)C(S_2) \leq C$ and $\alpha \text{Rev}(S_1)+(1-\alpha)\text{Rev}(S_2) \geq \text{OPT}_c$.

It is well known that any dynamic program can be formulated as a shortest or longest path problem on a directed acyclic graph. Further, the corresponding linear programming relaxations for these problems are well known to satisfy total unimodularity and have integral basic feasible solutions (see [30]). Therefore, for

both the nonparametric tree choice model and the sequential flips nonparametric choice model we can apply the linear programming based method to find S_1 and S_2 as well.

CHAPTER 7

CONCLUSION

In this thesis we introduced three variations of the nonparametric choice model in order to understand the practical settings in which this model can really shine.

First, we presented the nonparametric tree choice model. We begun by studying the assortment problem; we formulated this problem as a dynamic program in which the offer decision for each product can be made by simply storing each node's closest offered predecessor. Further, our dynamic programming formulation extended naturally to the cardinality constrained assortment problem and costed assortment problem. We then studied the joint assortment and pricing problem under special cases of the nonparametric tree choice model. Here, we developed novel dynamic programming approaches that additionally revealed nice structural results of the optimal pricing scheme. We concluded our analysis by providing a series of computational experiments that validated the use of the nonparametric tree choice model in practice. As shown, the sparsity of this model admits tractable estimation and optimization problems while capturing more complex customer behavior than the traditional multinomial logit choice model.

Next, we studied the k-product nonparametric choice model, which captures settings in which there is limited substitution between products. We proved that the assortment optimization problem under this model is NP-hard even for $k = 2$. Motivated by this result, we developed a 1.14-approximation algorithm for the assortment problem under the 2-product nonparametric choice model and a more general linear programming based approximation scheme whose approximation ratio decreases as k increases. Through a series of computational experiments, we showed that the performance of this latter algorithm far exceeds the theoretical

guarantees. We also showed that this model can efficiently capture substitution behavior for very small values of k, making the corresponding optimization algorithms efficient and robust to the choice of k.

The third model we introduced was the sequential flips nonparametric choice model. This model was motivated as a generalization of the linear utilities model, which is used for vertically differentiated products. Through this generalization, we were able to solve the assortment optimization problem using a simple dynamic program which extended to the joint assortment and pricing problem. Further, the dynamic program also extended easily to the cost and cardinality constrained versions of the problem. Finally, we presented a general two-stage approximation scheme for the space constrained assortment optimization problem that applied to two of our studied choice models.

One interesting future direction of work would be to consider the estimation and assortment problems for a mixture of nonparametric tree choice models or a mixture of sequential flips nonparametric choice models. In both cases, our dynamic programming ideas are rendered ineffective by adding this extra dimension. Another potential extension of our work would be to improve the approximation guarantees for the assortment optimization problem under the k-product nonparametric choice model for smaller fixed values of k. Additionally, it would be useful to consider the space constrained assortment problem under the k-product nonparametric choice model or the cardinality constrained assortment optimization problem for $k > 2$.

BIBLIOGRAPHY

[1] Gagan Aggarwal, Tomás Feder, Rajeev Motwani, and An Zhu. Algorithms for multi-product pricing. In Josep Díaz, Juhani Karhumäki, Arto Lepistö, and Donald Sannella, editors, *Automata, Languages and Programming: ICALP 2004*, Lecture Notes in Computer Science, vol 3142, pages 72–83. Springer, Berlin, Heidelberg, 2004.

[2] Ali Aouad, Vivek F. Farias, and Retsef Levi. Assortment optimization under consider-then-choose choice models. Available at SSRN: `https://ssrn.com/abstract=2618823`, 2015.

[3] Ali Aouad, Vivek F. Farias, Retsef Levi, and Danny Segev. The approximability of assortment optimization under ranking preferences. Available at SSRN: `https://ssrn.com/abstract=2612947`, 2015.

[4] Ali Aouad, Retsef Levi, and Danny Segev. Greedy-like algorithms for dynamic assortment planning under multinomial logit preferences. Available at SSRN: `https://ssrn.com/abstract=2655759`, 2015.

[5] Ali Aouad and Danny Segev. Display optimization for vertically differentiated locations under multinomial logit choice preferences. Available at SSRN: `https://ssrn.com/abstract=2709652`, 2015.

[6] Dimitris Bertsimas and Velibor Mišic. Data-driven assortment optimization. Working Paper, MIT Sloan School. Available at `https://www.algomus.com/wp-content/uploads/2016/11/Data-Driven-Assortment-Optimization.pdf` (accessed on May 25, 2017), 2016.

[7] Jose Blanchet, Guillermo Gallego, and Vineet Goyal. A Markov chain approximation to choice modeling. *Operations Research*, 64(4):886–905, 2016.

[8] Tudor Bodea, Mark Ferguson, and Laurie Garrow. Data set choice-based revenue management: Data from a major hotel chain. *Manufacturing & Service Operations Management*, 11(2):356–361, 2009.

[9] Niv Buchbinder and Moran Feldman. Deterministic algorithms for submodular maximization problems. In *Proceedings of the Twenty-Seventh Annual ACM-SIAM Symposium on Discrete Algorithms*, pages 392–403. Society for Industrial and Applied Mathematics, 2016.

[10] Niv Buchbinder, Moran Feldman, Joseph Seffi, and Roy Schwartz. A tight linear (1/2)-approximation for unconstrained submodular maximization. *SIAM Journal on Computing*, 44(5):1384–1402, 2015.

[11] John L. Crompton and Paul K. Ankomah. Choice set propositions in destination decisions. *Annals of Tourism Research*, 20(3):461–476, 1993.

[12] James M. Davis, Guillermo Gallego, and Huseyin Topaloglu. Assortment planning under the multinomial logit model with totally unimodular constraint structures. Working Paper, Department of IEOR, Columbia University. Available at `http://www.columbia.edu/gmg2/logit_const.pdf` (accessed on May 25, 2017), 2013.

[13] James M. Davis, Guillermo Gallego, and Huseyin Topaloglu. Assortment optimization under variants of the nested logit model. *Operations Research*, 62(2):250–273, 2014.

[14] James M. Davis, Huseyin Topaloglu, and David P. Williamson. Assortment optimization over time. *Operations Research Letters*, 43(6):608–611, 2015.

[15] James M. Davis, Huseyin Topaloglu, and David P. Williamson. Pricing problems under the nested logit model with a quality consistency constraint. *INFORMS Journal of Computing*, 29(1):54–76, 2016.

[16] Antoine Désir and Vineet Goyal. Near-optimal for capacity constrained assortment optimization. Available at SSRN: `https://ssrn.com/abstract=2543309`, 2014.

[17] Antoine Désir, Vineet Goyal, Srikanth Jagabathula, and Danny Segev. Assortment optimization under the Mallows model. In *Advances in Neural Information Processing Systems*, pages 4700–4708, 2016.

[18] Antoine Désir, Vineet Goyal, Danny Segev, and Chun Ye. Capacity constrained assortment optimization under the Markov chain based choice model. Available at SSRN: `https://ssrn.com/abstract=2626484`, 2015.

[19] Vivek F. Farias, Srikanth Jagabathula, and Devavrat Shah. A nonparametric approach to modeling choice with limited data. *Management Science*, 59(2):305–322, 2013.

[20] Uriel Feige, Vahab S. Mirrokni, and Jan Vondrák. Maximizing non-monotone submodular functions. *SIAM Journal on Computing*, 40(4):1133–1153, 2011.

[21] Jacob Feldman and Huseyin Topaloglu. Bounding optimal expected revenues for assortment optimization under mixtures of multinomial logits. *Production and Operations Management*, 24(10):1598–1620, 2015.

[22] Jacob Feldman and Huseyin Topaloglu. Capacity constraints across nests in assortment optimization under the nested logit model. *Operations Research*, 63(4):812–822, 2015.

[23] Guillermo Gallego, Anran Li, Van-Anh Truong, and Xinshang Wang. Approximation algorithms for product framing and pricing. Working Paper, Department of IEOR, Columbia University. Available at `http://www.columbia.edu/~vt2196/MPR_ORSubmission.pdf` (accessed May 25, 2017), 2016.

[24] Guillermo Gallego and Huseyin Topaloglu. Constrained assortment optimization for the nested logit model. *Management Science*, 60(10):2583–2601, 2014.

[25] Guillermo Gallego and Ruxian Wang. Multiproduct price optimization and competition under the nested logit model with product-differentiated price sensitivities. *Operations Research*, 62(2):450–461, 2014.

[26] Timothy J. Gilbride and Greg M. Allenby. A choice model with conjunctive, disjunctive, and compensatory screening rules. *Marketing Science*, 23(3):391–406, 2004.

[27] Michel X. Goemans and David P. Williamson. Improved approximation algorithms for maximum cut and satisfiability problems using semidefinite programming. *Journal of the ACM*, 42(6):1115–1145, 1995.

[28] John Hauser, Min Ding, and Steven P. Gaskin. Non-compensatory (and compensatory) models of consideration-set decisions. In *2009 Sawtooth Software Conference Proceedings, Sequin WA*, 2009.

[29] Dorit S. Hochbaum, Nimrod Megiddo, Joseph Naor, and Arie Tamir. Tight bounds and 2-approximation algorithms for integer programs with two variables per inequality. *Mathematical Programming*, 62(1):69–83, 1993.

[30] Alan J. Hoffman and Joseph B. Kruskal. Integral boundary points of convex polyhedra. In Michael Jünger, Thomas M. Liebling, Denis Naddef, George L. Nemhauser, William R. Pulleyblank, Gerhard Reinelt, Giovanni Rinaldi, and Laurence A. Wolsey, editors, *50 Years of Integer Programming 1958-2008: From the Early Years to the State-of-the-Art*, pages 49–76. Springer, Berlin, Heidelberg, 2010.

[31] Dorothee Honhon, Sreelata Jonnalagedda, and Xiajun Amy Pan. Optimal algorithms for assortment selection under ranking-based consumer choice models. *Manufacturing and Service Operations Management*, 14(2):279–289, 2012.

[32] M. Hosseinalifam, P. Marcotte, and G. Savard. Network capacity control under a nonparametric demand choice model. *Operations Research Letters*, 43(5):461–266, 2015.

[33] Norman Huang and Allan Borodin. Bounds on double-sided myopic algorithms for unconstrained submodular maximization. In *Proceedings of 25th International Symposium on Algorithms and Computation (ISAAC)*, pages 528–539, 2014.

[34] Srikanth Jagabathula. Assortment optimization under general choice. Available at SSRN: `https://ssrn.com/abstract=2512831`, 2014.

[35] Srikanth Jagabathula and Paat Rusmevichientong. A nonparametric joint assortment and price choice model. To appear, *Management Science*, 2016.

[36] Abel P. Jeuland. Brand choice inertia as one aspect of the notion of brand loyalty. *Management Science*, 25(7):671–682, 1979.

[37] Toshihiro Kamishima. Collaborative filtering: Recommendation based on order responses. In *Proceedings of the Ninth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 583–588, 2003.

[38] A. Gürhan Kök and Marshall L. Fisher. Demand estimation and assortment optimization under substitution: Methodology and application. *Operations Research*, 55(6):1001–1021, 2007.

[39] Eric Lapersonne, Gilles Laurent, and Jean-Jacques Le Goff. Consideration sets of size one: An empirical investigation of automobile purchases. *International Journal of Research in Marketing*, 49(1):55–66, 1995.

[40] Michael Lewin, Dror Livnat, and Uri Zwick. Improved rounding techniques for the MAX 2-SAT and MAX DI-CUT problems. In William J. Cook and Andreas S. Schulz, editors, *International Conference on Integer Programming and Combinatorial Optimization: IPCO 2002*, Lecture Notes in Computer Science, vol 2337, pages 67–82. Springer, Berlin, Heidelberg, 2002.

[41] Guang Li and Paat Rusmevichientong. A greedy algorithm for the two-level nested logit model. *Operations Research Letters*, 42(5):319–324, 2014.

[42] Guang Li, Paat Rusmevichientong, and Huseyin Topaloglu. The d-level nested logit model: Assortment and price optimization problems. *Operations Research*, 63(2):325–342, 2015.

[43] Sanjeev Mahajan and Hariharan Ramesh. Derandomizing approximation algorithms based on semidefinite programming. *SIAM Journal of Computing*, 28(5):1641–1663, 1999.

[44] Siddharth Mahajan and Garrett van Ryzin. Inventory competition under dynamic consumer choice. *Operations Research*, 49(5):646–657, 2001.

[45] Siddharth Mahajan and Garrett van Ryzin. Stocking retail assortment under dynamic consumer substitution. *Operations Research*, 49:334–351, 2001.

[46] Isabel Méndez-Díaz, Juan José Miranda-Bront, Gustavo Vulcano, and Paula Zabala. A branch-and-cut algorithm for the latent-class logit assortment problem. *Discrete Applied Mathematics*, 164:246–263, 2014.

[47] Michael Mussa and Sherwin Rosen. Monopoly and product quality. *Journal of Economic Theory*, 18(2):301–317, 1978.

[48] George L. Nemhauser and Leslie Earl Trotter. Vertex packings: structural properties and algorithms. *Mathematical Programming*, 8(1):232–248, 1975.

[49] Xiajun Amy Pan and Dorothée Honhon. Assortment planning for vertically differentiated products. *Production and Operations Management*, 21(2):253–275, 2012.

[50] Matthias Poloczek, George Schnitger, David P. Williamson, and Anke van Zuylen. Greedy algorithms for the maximum satisfiability problem: Simple algorithms and inapproximability bounds. To appear, *SIAM Journal on Computing*, 2016.

[51] W. Zachary Rayfield, Paat Rusmevichientong, and Huseyin Topaloglu. Approximation methods for pricing problems under the nested logit model with price bounds. *INFORMS Journal of Computing*, 27(2):335–357, 2015.

[52] Paat Rusmevichientong, Zuo-Jun Max Shen, and David B. Shmoys. Dynamic assortment optimization with a multinomial logit choice model and capacity constraint. *Operations Research*, 58(6):1666–1680, 2010.

[53] Paat Rusmevichientong, David B. Shmoys, Chaoxu Tong, and Huseyin

Topaloglu. Assortment optimization under the multinomial logit model with random choice parameters. *Production and Operations Management*, 23(11):2023–2039, 2014.

[54] Paat Rusmevichientong and Huseyin Topaloglu. Robust assortment optimization in revenue management under the multinomial logit choice model. *Operations Research*, 60(4):865–882, 2012.

[55] Paat Rusmevichientong, Benjamin Van Roy, and Peter W. Glynn. A nonparametric approach to multiproduct pricing. *Operations Research*, 54(1):82–98, 2006.

[56] Kalyan Talluri and Garrett van Ryzin. Revenue management under a general discrete choice model of consumer behavior. *Management Science*, 50(1):15–33, 2004.

[57] Kenneth Train. *Discrete Choice Models with Simulation*. Cambridge University Press, 2009.

[58] Garrett van Ryzin and Gustavo Vulcano. A market discovery algorithm to estimate a general class of nonparametric choice models. *Management Science*, 61(2):281–300, 2015.

[59] Garrett van Ryzin and Gustavo Vulcano. Technical note—an expectation-maximization method to estimate a rank-based choice model of demand. *Operations Research*, 65(2):396–407, 2017.

[60] Ruxian Wang. Capacitated assortment and price optimization under the multinomial logit model. *Operations Research Letters*, 40(6):492–497, 2012.

[61] Ruxian Wang. Assortment management under the generalized attraction model with a capacity constraint. *Journal of Revenue and Pricing Management*, 12(3):254–270, 2013.